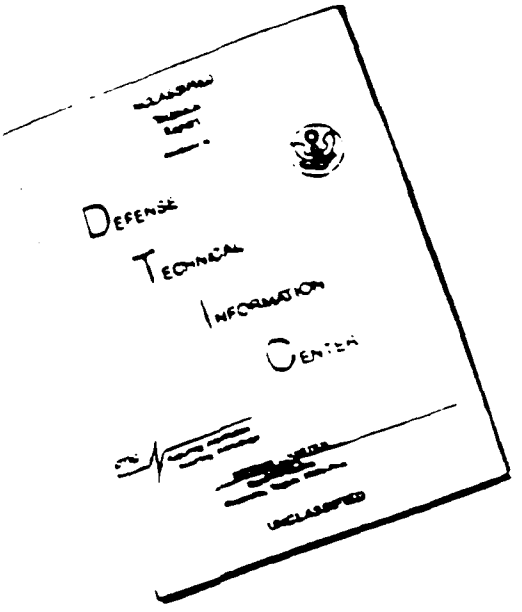
OBJECT TRACKING THROUGH ADAPTIVE CORRELATION

THESIS

Dennis Anthony Montera
Captain, USAF

AFIT/GEO/ENG/92D-07

DISCLAIMER NOTICE



THIS DOCUMENT IS BEST QUALITY AVAILABLE. THE COPY FURNISHED TO DTIC CONTAINED A SIGNIFICANT NUMBER OF PAGES WHICH DO NOT REPRODUCE LEGIBLY.

AFIT/GEO/ENG/92D-07

OBJECT TRACKING THROUGH ADAPTIVE CORRELATION

THESIS

Presented to the Faculty of the School of Engineering
of the Air Force Institute of Technology
Air University
In Partial Fulfillment of the
Requirements for the Degree of
Master of Science in Electrical Engineering

Dennis Anthony Montera, B.S.E.E.
Captain, USAF

17 December, 1992

Approved for public release: distribution unlimited

Acknowledgments

I would like to thank my advisor, Maj. Steven K. Rogers, along with the rest of my thesis committee, Capt. Dennis W. Ruck and Dr. Mark Oxley, for giving me the opportunity to tackle this problem. I would especially like to thank Maj. Rogers for his guidance, insight, and encouragement which allowed me to succeed while also allowing me to find my own path.

I also want to thank the people at the Model Based Vision Laboratory for the assistance they gave me. I would especially like to thank Mr. Jim Leonard, who took the time to listen to my ideas, offered suggestions, and gave me insights into other projects being worked in this area.

Most importantly, I would like to thank my wife, Ilona, for bearing with me when times were tough, and enjoying life with me when we had a little free time. Her patience and understanding with the hours I spent working on this research helped more to make it possible than she will ever know.

Dennis Anthony Montera

Table of Contents

4

List of Figures

List of Tables

Abstract

This paper discusses the use of a correlation based system to track an object through a series of images based on templates derived from previous image frames. The ability to track is extended to sequences which include multiple objects of interest within the field of view of the correlator system. This is accomplished by comparing the height and shape of the correlation of the template correlated with itself to the height and shape of all peaks in the correlation of the template with the next scene. The result is to identify the region in the next scene which best matches the shape and intensity of the designated target. The use of correlation plane information for the designation of other objects of interest within the field of view will also be demonstrated. This process compares only the shape of the template correlated with itself to the shape of all peaks in the correlation of the template with the next scene. The result of this process is to identify all regions in the next scene which closely resemble the shape of designated target. In addition to correlation plane postprocessing, an adaptive window is used to determine the template size in order to reduce the effects of correlator walk-off. Two image sequences were utilized in the conduct of this research. These sequences, taken from a Forward Looking Infrared (FLIR) sensor mounted onboard a DC-3 aircraft, contain a T-55 tank and both an M-113 and a TAB-71 armored personnel carrier moving in a columnized formation along a dirt road. Results indicated that postprocessing of correlation plane information could discriminate between multiple targets in the sensors field of view (FOV), and maintain a track on the designated target. Postprocessing of correlation plane information also indicated that other targets within the FOV could be located.

OBJECT TRACKING THROUGH ADAPTIVE CORRELATION

I. Introduction

The ability to track an object autonomously in a cluttered environment with several objects within a sensor's field-of-view (FOV), has been a goal of both the military and industry for some time. While the potential applications of such a system are unlimited in industry, the military is investigating a capability which would allow a warhead to track a target with minimum human interaction. For the purpose of this research, the ability to track a target is achieved when an electro-optic sensor system is able to correctly determine the location of a target within its FOV through time as both the sensor and the target are moving. In order for such a system to be effective, the tracker must be able to maintain lock on a target despite clutter, a changing target signature, multiple targets within the FOV, and partial obscurations. The ability to ultimately identify the target, and possibly change lock to another, higher value target within the FOV is also desired. Finally, the system must be capable of operating in real-time to be of any military use. Although many tracking systems have been developed to date, none meet all of these strict requirements at all times.

1.1 Background

The battleground of the future continues to become more and more complex, greatly increasing in complexity the demands placed on the pilots of high performance aircraft and helicopters. One means of reducing tasks placed on these pilots is to develop a real-time, autonomous tracking system, capable of locking onto targets in a cluttered, highly dynamic battlefield. Any such tracking system must be capable of handling a large set of potential targets; targets which may vary in size,

10

shape, and intensity. To maintain low cost the tracking system must be compatible with current airborne sensor systems such as the Forward-Looking Infrared (FLIR) sensor. Finally, the system must attain track quickly in order to ensure timely delivery of onboard weapon systems.

The areas of autonomous image tracking and identification have been investigated rigorously by students at the Air Force Institute of Technology (AFIT) under the direction of Dr. Steven Rogers. These studies have examined many potential methods and algorithms for achieving autonomous image tracking (10) (14).

One potential system researched was the correlation based tracker, investigated by Capt. Paul Law, GE 91-D, in his thesis titled "Correlation Based Distortion Invariant Infrared Tracking" (10). Correlation algorithms measure the similarity between a target template and the actual scene. To avoid the need to store large numbers of templates representing all possible orientations of all possible targets, Capt. Law implemented an adaptive correlator which generated its template from the last frame of the actual scene. With the adaptive correlation algorithm, Capt. Law was able to maintain track of a designated target using real FLIR imagery which contained multiple possible targets and large amounts of background clutter. Because correlations can be computed optically through Fourier Optics techniques at nearly the speed of light, a real-time autonomous tracker based on optical, adaptive correlations was projected as an achievable system.

1.2 Problem Definition

An autonomous, real-time image tracking system which can identify targets and choose highest value targets among multiple targets in its FOV does not currently exist. Any such system must be capable of correctly tracking the position of a designated target despite such problems as a cluttered FOV, multiple targets within its FOV, and changes in the position and orientation of both target and tracker.

Furthermore, it must be capable of identifying targets and choosing those with the highest priority or target value.

1.3 Scope and Limitations

Much of the recent work in tracking has concentrated on developing algorithms which predict the possible locations of a target, then choose the the target out of all detected objects which best matches the projected location (1) (2) (3) (7) (11) (16) (18). These research efforts concentrated on determining which detected objects were the targets being tracked, and not on how to find potential targets. One potential method of finding and tracking targets at the same time is through a correlation based tracker. Southern Research Technologies, Inc. has developed a real time correlation based system which tracks a designated target in a cluttered environment (17). However, this system searches only a small area of the entire scene, and multiple targets of similar shape and size are not present in the FOV..

This research was a continuation of that conducted by Capt. Paul Law and was intended to improve upon the methods he developed while introducing some new capabilities. Capt. Paul Law demonstrated in his thesis work that a correlation based system was capable of tracking a designated target in a cluttered, multi-target environment while both the target and the tracker were moving (10). The main goals of this research were to: 1) reduce the tracking error encountered by Capt. Law in his research (10), 2) expand the portion of scene used to determine if other possible targets are present, 3) to track multiple targets if present, and 4) to address the issue of correlator "walk-off" and identify solutions to reduce this effect. The ability to identify all potential targets being tracked, and to choose the highest value target present and change lock to that target were left open for future research. To reduce the amount of effort necessary to achieve these objectives, the following limitations were imposed.

- The task of target acquisition (initially detecting and designating the target) was not addressed. It was assumed that initial target designation was achieved manually by an operator, and that the system need only maintain track of the target once this was accomplished.
- Computer simulations using real FLIR images were conducted only, and thus the system did not operate in real-time. However, it was felt that an optical implementation of the system modeled would be capable of functioning in real-time.

1.4 Methodology

Along with improving the performance achieved by Capt. Law (10), this effort focused on addressing the issue of correlator walk-off, using the information in the template auto-correlation peak, and the capability to track multiple targets within the tracker's FOV.

In research conducted by Tam (13), an accumulated tracking error was encountered. While the optical correlator used in his research may have introduced tracking nonlinearities (6) and random noise (4) errors, this thesis investigated "walk-off" error inherent in an adaptive correlation based tracker, identified its sources, and tested solutions for reducing its effects.

In a scene with multiple targets present, it is possible that each target will yield a peak through the correlation process. Each of these peaks will vary in amplitude, and some may be larger than the peak corresponding to the target being tracked. Therefore, a method of using the information contained in the shape of the correlation peak (14) was introduced. In this method, the shape of the correlation peak of the template correlated with itself was used to identify the template-scene correlation peak which most closely resembled it. This peak was then determined to be that which corresponds to the target being tracked.

Since Capt. Law had no means of distinguishing the correlation peak of the target being tracked from the peaks corresponding to other objects in the FOV, he reduced the area of the scene frame searched to exclude other possible targets (10). However, since the method of using correlation peak shape made it possible to identify one target from all others within the FOV, it was now possible to open the scene searched to include the entire frame. This allowed for the tracking of all other secondary targets within the tracker's FOV.

Computer simulations for this research were conducted on the SUN Sparc 2 workstations using the KHOROS image and signal processing software developed by the faculty and students at the University of New Mexico (9). The FLIR images used were those utilized by Capt. Law (10) which were provided by the Model-Based Vision Laboratory, WL/AARA, Wright-Patterson AFB, Ohio. These images were 499x320 pixels in size with 256 possible grey scale values per pixel. The tracking algorithms used were modifications of the programs written by Capt. Law in the C programming language.

1.5 Overview of Thesis

This thesis is organized into five chapters and an appendix. The order of chapters represents the chronological evolution of this research effort from an idea to theory, implementation to results, and conclusions. The appendix contains copies of the C language programs used to conduct the adaptive correlation process.

This chapter introduces the thesis topic, while also presenting the scope and methodology implemented in this research. Chapter II discusses the basic concepts of correlation and adaptive correlation as they relate to this effort. Chapter III describes the evolution of and procedures used to conduct the computer simulations of adaptive correlation tracking. Chapter IV discusses and analyzes the results obtained during this research. Chapter V presents the research conclusions and discusses potential research areas for future efforts.

II. Basic Concepts

In his thesis (10), Capt. Paul Law provided a detailed review of the concept of correlation, adaptive correlations, their uses, and possible optical implementations of a correlator system. Also, Kumar (8) provides an overview of various optical correlator designs, including both strengths and weaknesses of each. This chapter briefly discusses the fundamentals of correlation and describes what is meant by adaptive correlations.

2.1 Correlation

Correlation is a mathematical process which measures the similarity between two objects. The correlation of `s(x,y)` and `t(x,y)` when both are real is

$$s(x,y) \star t(x,y) \triangleq \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} s(\alpha,\beta)\, t(\alpha - x\,\beta - y)\, d\alpha\, d\beta \qquad (1)$$

or

$$s(x,y) \star t(x,y) = \mathcal{F}^{-1}\left[S(\xi,\eta) \cdot T^{*}(\xi,\eta)\right] \qquad (2)$$

the template is positioned at every possible location in the field of the search scene. For each possible location, the two images are multiplied, and the value obtained represents the similarity, or correlation, of the two images at those locations. The correlation should be large when the two objects overlap, and small or zero when no overlap occurs. When the two objects are at the location where the best match between the two occurs, a peak in the correlation plane is expected.

There are, however, some factors which affect the results of a correlation. First, if the object in the search scene varies greatly in size, shape, or orientation from the template, no true correlation peak may occur and the object in the search scene may

fade into the background noise. Also, areas in the search scene which have larger or brighter values than those of the object to be located, may produce larger correlation peaks than those created by the true object of interest. These are factors which must be accounted for in any correlation based tracker.

2.2 Adaptive Correlation

Adaptive correlation applies to the desire to track an object while the scene and object being tracked are changing with time. Since the object's size, shape, and orientation may be changing, the template being used must adapt for a correlator to maintain a good track. This is accomplished by continually updating the template used in the correlation process. Each time the target is identified in the search scene, the object is extracted from that scene to be used as the template for the next frame of the sequence. Although little research has been reported in the area of correlation post processing (8), analysis of the correlation plane information with and without energy normalization proved itself to be very valuable in distinguishing between multiple targets within the sensors FOV. As long as the time between frames is not large enough to allow extreme changes in size, shape, or orientation of the object being tracked, this method of adapting the template to the current scene should be sufficient to allow correlation based tracking. An example scene, template, and correlation plane are shown in Figure 1.

2.3 Summary

This chapter briefly reviewed the basic concepts necessary for the conduct of this thesis. An overview of correlations and the concept of adaptive correlations were presented, and the reader is referred to the thesis of Capt. Law (10) for a more in depth discussion of these topics.

Chapters III and IV describe the methodology used and results obtained using an adaptive correlation based system to track real targets in FLIR generated imagery.

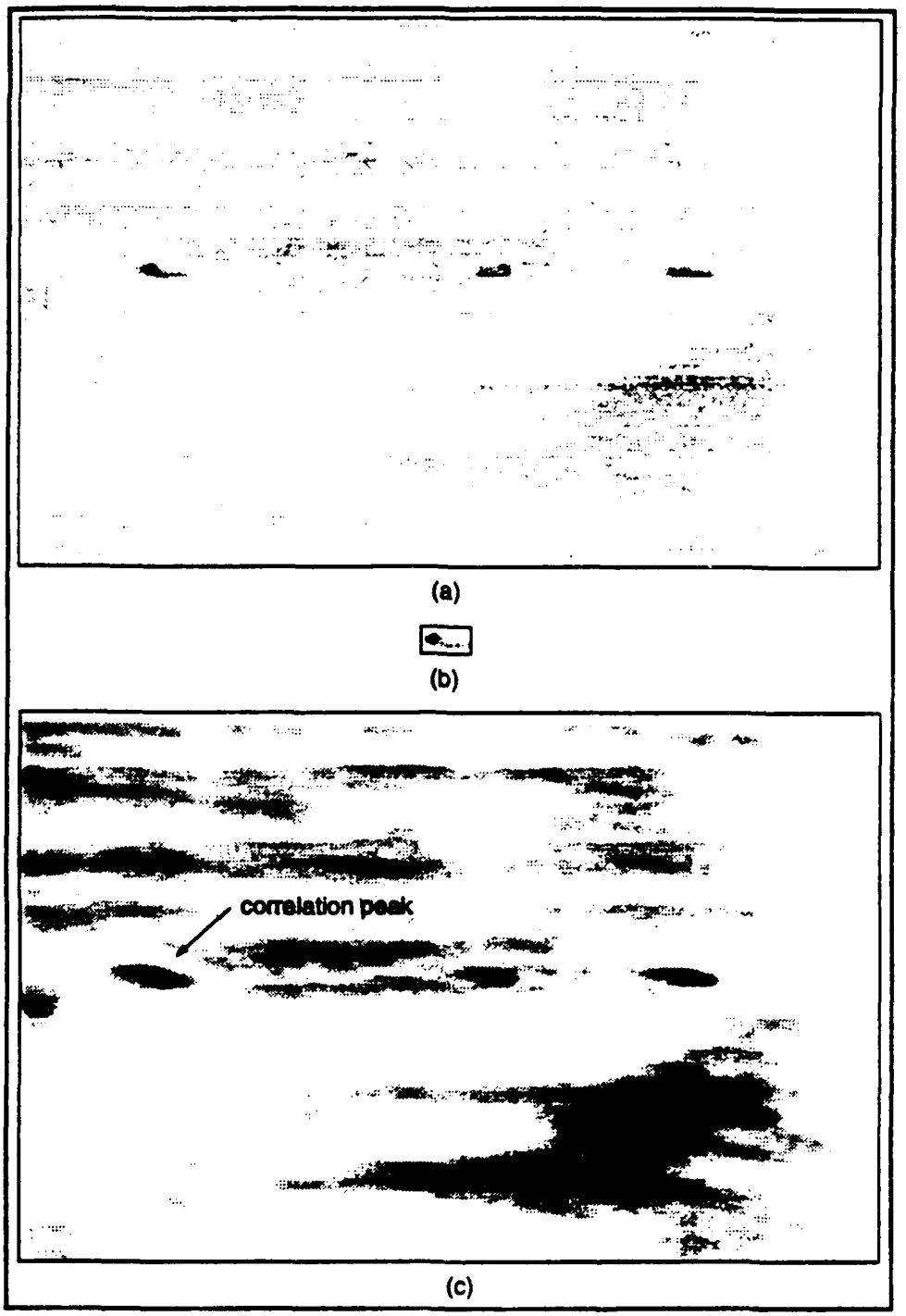


Figure 1. Example correlation shown in negative. (a) scene, (b) template, and (c) correlation plane

III. Methodology

This chapter discusses the methodology and approach taken for this experiment in adaptive correlation based tracking. In his thesis (10), Capt. Law provided a very detailed discussion of the hardware, software, images, and correlation method used during his research. A brief review of this information is given, followed by a detailed presentation of the changes made to expand and improve the performance of the correlation based tracking system.

3.1 Hardware and Software

All programming and simulations for this research were conducted on Sun Sparc 2 workstations. Each workstation operated at 28 MIPS, was equipped with 32 MBytes of internal memory, and the stations were networked together via thin wire ethernet.

Programs written in the C programming language by Capt. Law (10), were modified for this research. The programs, called `TRAC_MENU.C` and `TRAC_PRGM.C` are found in Appendix B.1 and B.2. These programs utilized routines of the KHOROS image processing software (9) developed by the faculty and students at the University of New Mexico. While some processing was done within the C programs, their main purpose was to control the sequencing and flow of routines within KHOROS to conduct the adaptive correlation process.

3.2 FLIR Images

Two FLIR image sequences were utilized in this research. These images were taken from a Helicopter Fire Control System (HFCS) FLIR mounted on board a DC-3 aircraft. These sequences were taken at the Aberdeen Proving Ground, Maryland, between October 21 and December 4, 1982, between the times of 2000 and 2200

hours. The targets consisted of a T-55 tank and both an M-113 and a TAB-71 armored personnel carrier moving in a columnized formation along a dirt road.

In one sequence the aircraft flew toward the targets perpendicular to their movement along the road, and in the other the aircraft flew in parallel and from behind the formation. In both sequences the aircraft flew at approximately 1000 feet of altitude and an operator electro-mechanically steered the sensor to track the targets from a range of approximately 10 kilometers to 1 kilometer distance. The FLIR images were provided by the Model-Based Vision Laboratory (WL/AARA).

The images provided were 499x320 pixels in size with 256 gray scale values available. These images represented approximately 1 second between frames, or approximately 65 meters of flight distance per frame. In both sequences the targets were warmer than the surrounding background, and thus had higher pixel values than the background data. This made the targets distinguishable by the human eye from a range of approximately 7 kilometers. In the case where the plane flew parallel to the path of the targets, all three targets appeared close together and remained within the sensors FOV for the entire sequence. However, true target location information was only available down to 2200 meters for Target 3. Therefore, all tracking information gathered on Target 3 for this sequence ended at this range. In the case where the aircraft flew perpendicular to the three targets, the separation between the vehicles made it impossible to keep all three within the sensors FOV throughout the sequence. At approximately 2700 meters one target exited the field of view, while another exited the FOV at approximately 1900 meters. The final target, the M-113 armored personnel carrier, remained in the FOV for the entire sequence. Example frames from Sequence 1 are shown in Figure 2 and from Sequence 2 in Figure 3.

For both sequences, information about the true target locations for all targets was provided by the personnel at the MBV Laboratory for frames between the ranges of 8 to 1 kilometers. This data was obtained by analyzing each frame and finding the

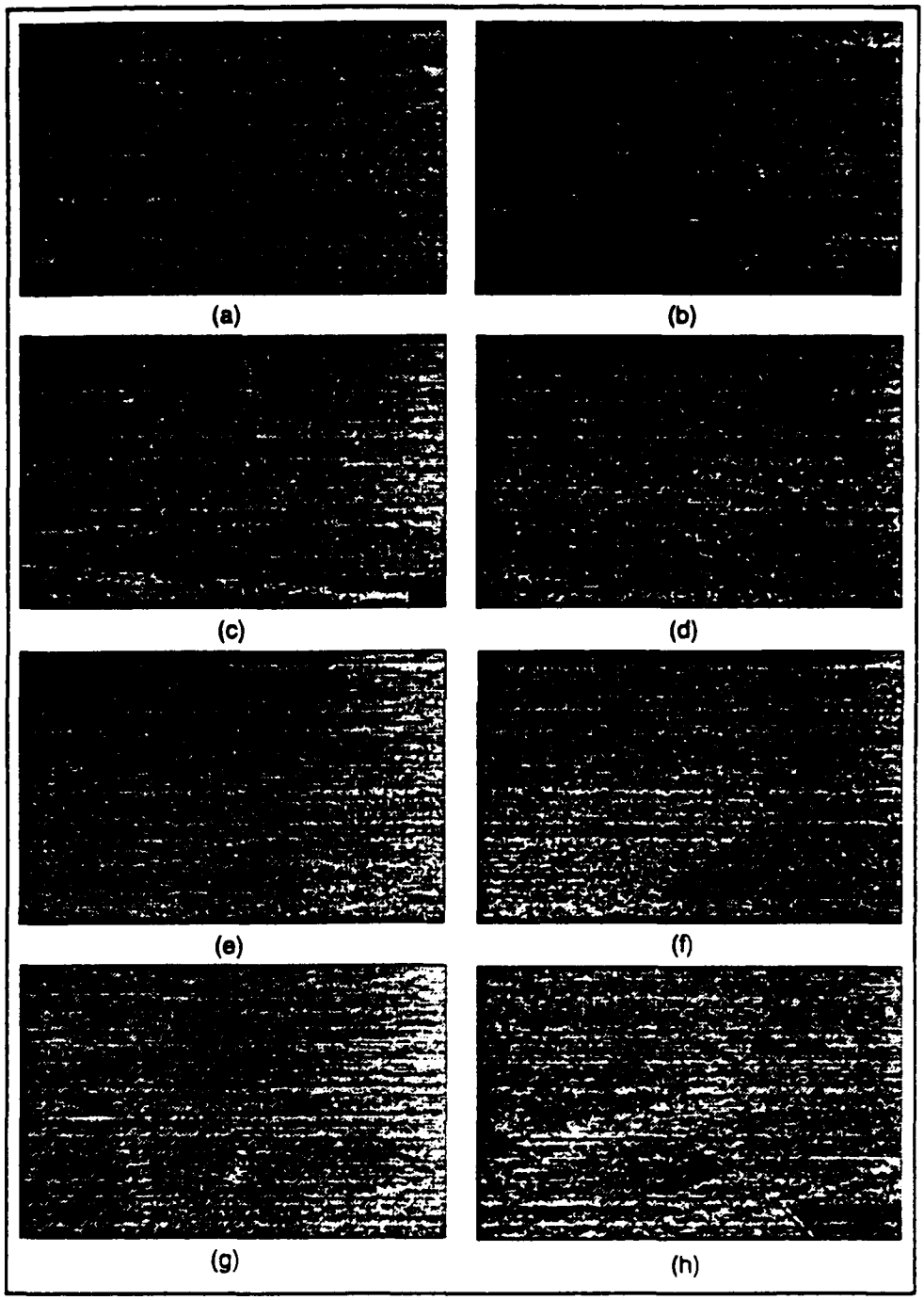


Figure 2. Sample frames from Sequence 1 shown in negative. Images represent ranges from 8 km (a) to 1 km (h).

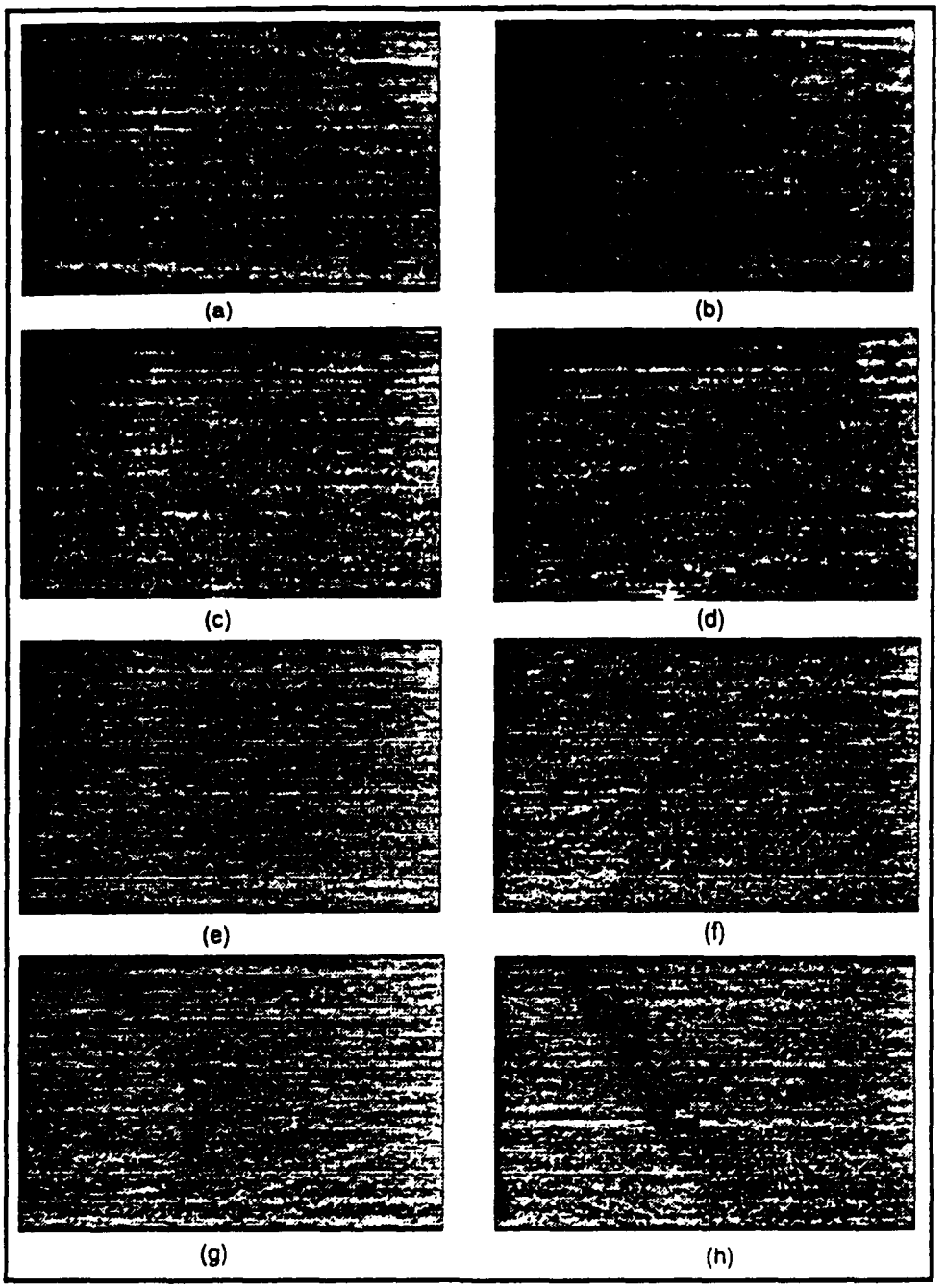


Figure 3. Sample frames from Sequence 2 shown in negative. Images represent ranges from 8 km (a) to 1 km (h).

bounding box which encompassed each target entirely. The center of each box was considered to be the center of the target. This information was used to determine the error in numbers of pixels that the correlation based tracker achieved for each frame.

3.3 *Correlation Process*

The adaptive correlation process as implemented by the C language programs was discussed in full detail in Capt. Law's thesis (10), and a brief, less detailed overview is given in this section. This process remained unchanged during this research except as noted in later sections of this chapter.

The process began by loading the previous and current frames of the sequence. Both images were then preprocessed based on the operator's selection of built in options. The template was then removed from the previous frame, and the search scene was extracted from the current scene, with their sizes dependent on the operator's selected options. The two images were then correlated, and the resulting peak in the correlation plane was chosen as the location of the target. This location was then used as the center for the extraction of the template to be used for the next frame. This process repeated itself and continued through the entire sequence of images.

For the size of the template and the search scene, the operator had three options in the original version of the program (10). The sizes were: small (template 30x20, scene 60x38), medium (template 42x28, scene 116x72), and large (template 64x40, scene 192x120). These sizes were fixed throughout the correlation process. During this research two new options were added. For both, the template became adaptive. That is, the program tried to determine the exact size of the target, and make the window for the template that same size. In one case, the scene was 60x38, the same as the original small scene, and in the other case, the search scene became the entire 499x320 frame. During this research, only the two new options were utilized to compare the results obtained to those obtained by Capt. Law (10).

The programs also contained many possibilities for image preprocessing. Examples of these options included: none, threshold binarization, threshold binarization with gray scale restored, image blending, and other combinations. An option to use an edge extractor (9) after implementing any of the previous options was added during this research. Sample images are shown in Figure 4.

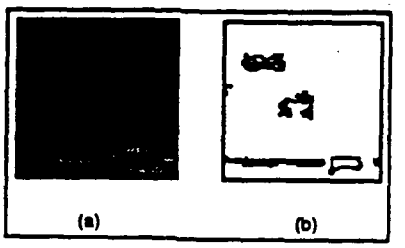


Figure 4. Example image before and after edge extraction (9) shown in negative. (a) original image, (b) image after edge extraction.

Under the option of threshold binarization, three possibilities were implemented. These are: scene average thresholding, line-by-line thresholding, and Cline thresholding. Scene average thresholding set to 0 all pixels in the scene that had values less than the scene average plus an operator set percentage, while setting all pixels above that value to 1. Line-by-line thresholding utilized the average of the eight previous pixels plus an operator set percentage to determine if a pixel was set to 0 or 1. Cline thresholding made its decision based on the average plus an operator set percentage of the eight pixels surrounding the pixel of interest. Based on the results obtained by Capt. Law (10), an operator set percentage of 15% was always chosen when a binarization technique was utilized. Once binarization had taken place, grey-scale was restored if the operator desired. This was accomplished by multiplying pixel-by-pixel the binarized image with the original image. Therefore, all pixels set to zero through binarization remained zero, while all pixels set to one through binarization were restored to their original grey-scale values. A flow diagram of the final program is shown in Figure 5.

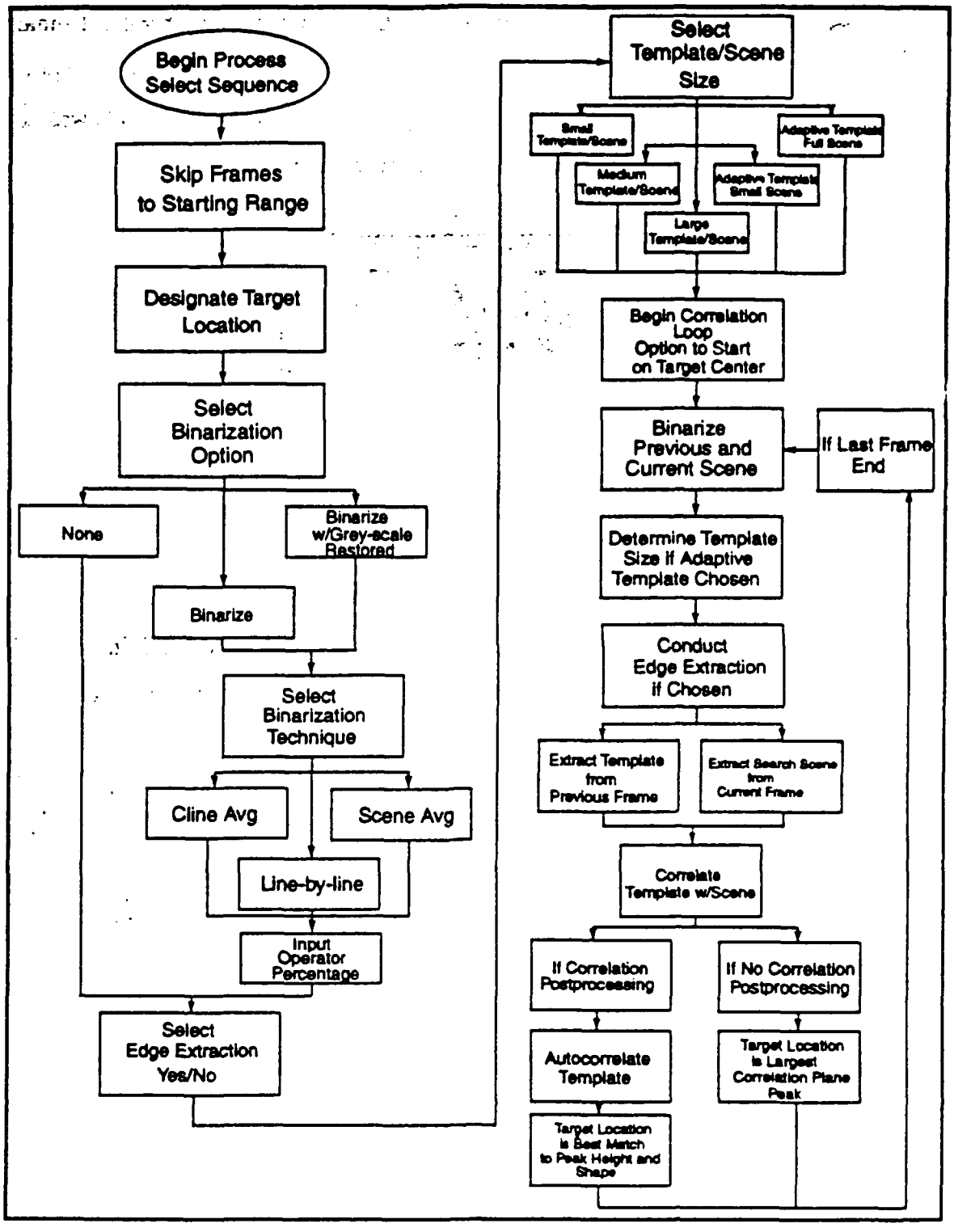


Figure 5. Flow diagram of the correlation based tracker program.

During this research effort, only the following options were used: no image preprocessing, no binarization with edge extractor, and scene average, line-by-line, and Cline thresholding all with gray scale restored. The three binarization techniques were all implemented to determine which worked best with the adaptive template, and which generated the lowest average tracking error. Then, the best binarization technique was compared to no preprocessing, and no binarization with edge extraction to determine which image preprocessing technique provided the best overall tracking performance.

3.4 Walk-off and Error Reduction

Whenever the object of interest is changing its size, shape, or orientation from frame to frame of a sequence, correlator walk-off will occur. When the template does not exactly match the target in the search scene, the point where the template best matches the target may not correspond with the center of the target. This produces an error in the correlation. Although this error is a random error, it will continue to add to itself from frame to frame due to the nature of the correlation process. Given enough frames, this error may eventually accumulate to such a point that it "walks-off" from the target (13).

Also, it was found in Capt. Law's research (10), that the larger the template was, the more error that was introduced into the correlation. This was due to the background pixels which were a part of the template when the template was much larger than the target. Just as variations in the target from frame to frame introduced error, so did the background pixels. By eliminating as much of the background information from the target as possible, the error from frame to frame could be reduced.

The solution to both of these problems became an adaptive window for creating the template. A windowing algorithm that found the edges of the target offered several advantages. First, the target location as determined through correlation was

assumed to be a pixel in the area of the target, but not necessarily the center of the target. After the target had then been windowed by a bounding box, the center of that bounding box was considered to be the true center of the target. By doing this, the system compensated for the walk-off error encountered from frame to frame, and reduced the effects of its accumulation. At the same time, the adaptive window reduced the error encountered each time by setting the size of the template to the size of the target and thus removed excess background pixels which had an impact on the correlation process.

An added benefit from an adaptive template was in the area of target designation. Without the adaptive template, if the original pixel location for the target as designated by the operator was not the true center of that target, error was introduced into the system. Capt Law's solution (10) was to allow the operator to select the true center of the target as a starting point because this information was available to him. Although this allowed him to begin the tracking process without an operator introduced error, it did not represent the real world where a target designator does not know the exact center location of the target. However, the adaptive template option did allow the operator to designate any pixel of the target as a starting location, and by bounding the target to select the template, operator introduced error was reduced if not eliminated.

3.5 Adaptive Template

The algorithm developed to perform the adaptive windowing was based on previous knowledge of the content of the sequences. Its intent was to validate the concept and performance improvements that could be attained by having an adaptive template, not to validate the algorithm developed. Development of a true adaptive template algorithm which would work for a variety of situations was left as a topic for future research.

Two pieces of prior knowledge allowed the algorithm developed in this research to work. First was that most of the background pixels had values below 70. Second was that all of the targets were brighter than the background, and that the majority of the target pixels had values greater than 70. By utilizing this information, an adaptive template algorithm was developed.

The algorithm began by checking the location of the correlation peak to see if the pixel value in the current scene at that location was greater than 70. If it was, the algorithm assumed that it was part of the target. If not, it assumed the pixel was near the target and began an outward search for the target. This search included checking outward in all directions from the current location until the nearest pixel with a value greater than 70 was found. This pixel was then considered to be part of the target.

Once a pixel was determined to be part of target, a 3x3 box was drawn around it. Each side of the box was then checked to see if 95% of its pixels had values less than 70. If all sides of the box had 95% of their pixels less than 70, the windowing would be complete and the size of the template determined. If not, each side of the box that did not meet the 95% criteria was pushed out from the starting location of the window. This increased the size of the window, but only expanded it in the directions of the edges of the target from the starting location. Again all four sides were then checked to see if they met the 95% criteria. Expansion of the window continued until all four sides met the specified criteria. At that point, the window matched the actual size of the designated target, and the center of the window was then considered to be the true center of the target. The size of the template utilized for the next frame was also set to the size of the adaptive window. Figure 6 provides an example of the process. This figure demonstrates the whole process of starting with a pixel off of the intended target, to finding the nearest target pixel, to expanding the window until all sides meet the 95% criteria. Figure 7 shows examples of adaptive templates taken with various preprocessing options.

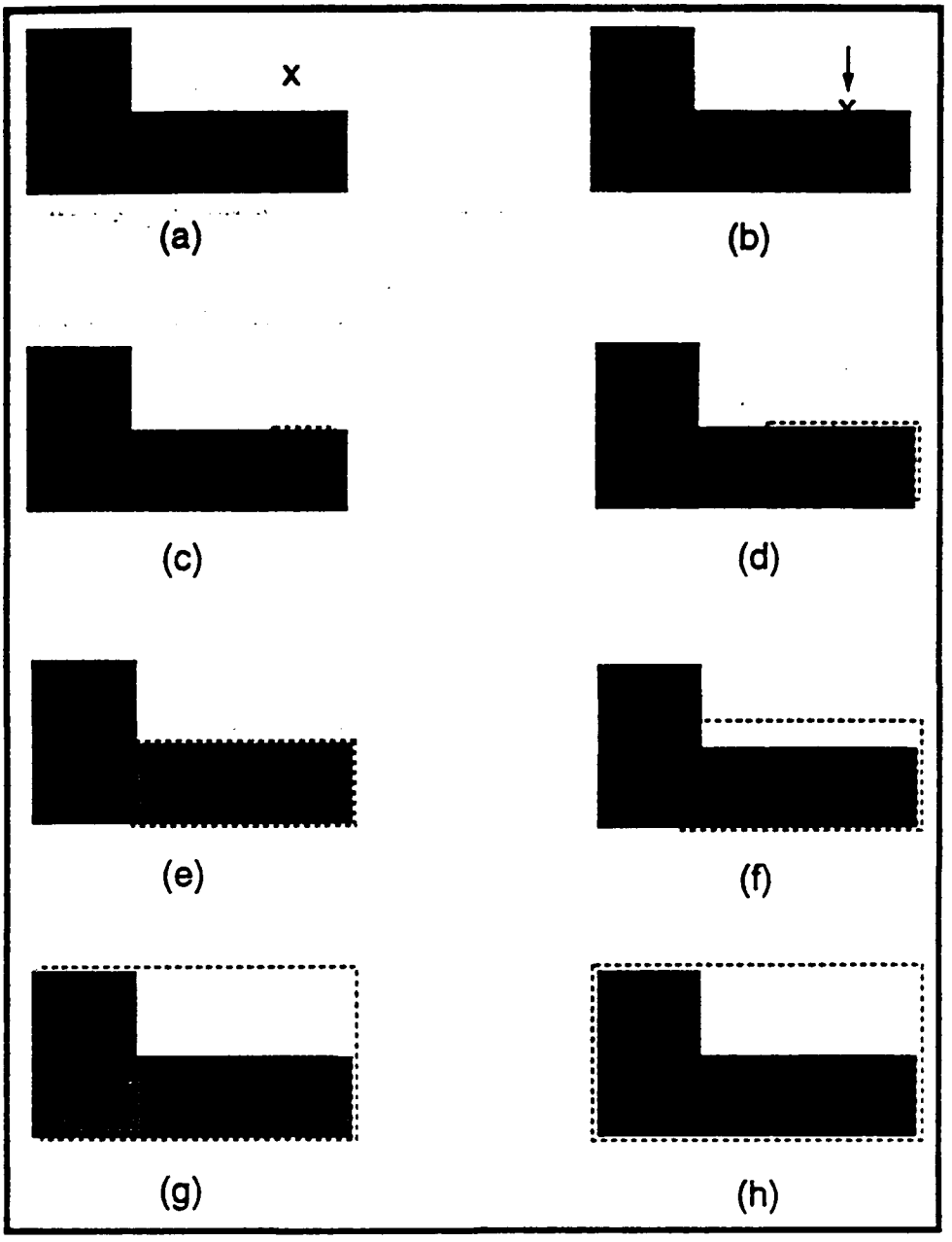


Figure 6. Adaptive template windowing process. (a) correlation peak location not on target, (b) process finds nearest target pixel, (c-g) window expands in each direction that contains greater than 95 percent target pixels, and (h) window process encompasses entire target.

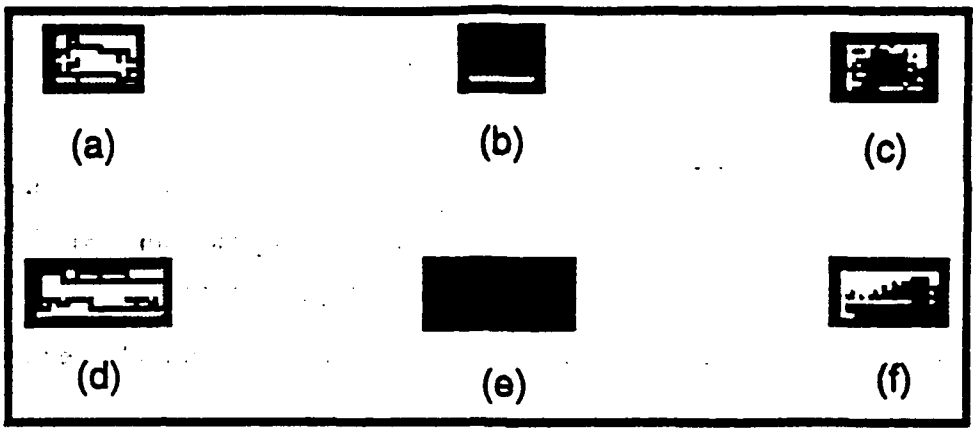


Figure 7. Sample templates from (a-c) Sequence 2 and (d-f) Sequence 1 shown in negative. (a,d) show edge extraction, (b,e) represent no image preprocessing, and (c,f) represent scene averaging with grey-scale restored.

3.6 Distinguishing Between Multiple Targets

In his research (10), Capt. Law restricted the search scene for each frame to a subimage of the full frame. The size of the subimage was based on the option of small, medium, or large selected by the operator. His research found that the smaller the search scene, the better the performance of the system. The reason the system worked better with a small search scene was because it reduced the number of targets in the search scene. As mentioned in Chapter 2, when multiple targets of differing brightness are in the search scene, problems may be encountered when trying to track a target which is not the brightest. This problem occurred because the brightness of pixels can outweigh the match in shape in the correlation process, resulting in the highest correlation peak on the brightest target. This caused the correlator to shift its track from the designated target to the brightest target in the FOV.

The shift in tracking to the brightest target produced a problem in how to maintain track on targets which are not the brightest target in the FOV. Analysis of the correlation plane showed that although the brightest target produced the largest

peak, other, smaller peaks did exist at the locations of the other targets and at many locations in the background. Therefore, the problem became how to identify the proper peak in the correlation plane, not just the tallest peak. Postprocessing of the correlation plane information yielded promise in solving this problem. However, little has been reported in the area of correlation postprocessing (8). The research that has been reported (12) (14) (15) has all dealt with identifying a target that has been scaled or rotated, and not in distinguishing between multiple targets of the same shape.

3.6.1 Template Autocorrelation When the amount of time between frames in a sequence is small enough that the objects being tracked do not change much in size, shape, or orientation, then the template holds more information than the shape and brightness of the target. Since the target in the current scene was expected to look similar to the object in the template, then the size and shape of the template correlated with the target was expected to look similar to the size and shape of the template correlated with itself. Taking advantage of this information became the focus of the rest of this research effort.

However, it should be noted that the template autocorrelation utilized was not a true autocorrelation. If the template had only been correlated with itself, then as soon as the template was shifted off center, the energy in the correlation plane would drop dramatically. This would not be representative of what happened when the template was correlated with the scene because the background surrounding the target was not being considered in the shape of the correlation peak. Therefore, two templates were extracted. The first template was the true template which was to be correlated with the next scene. The second template was a copy of the first template, only expanded an additional three pixels in each direction. When it came time for template autocorrelation, the true template was correlated with the expanded template. This allowed the background surrounding the template to influence the correlation plane information. The resulting correlation plane peak's height and

shape were then more representative of what was expected when the true template was correlated with the next scene.

3.6.2 Correlation Peak Identification In his thesis (14), Troxel investigated the use of correlation information for pattern recognition of targets. Because he experienced some success with his process, the methods utilized by Troxel became the starting point in this research for identifying the proper peak in the correlation plane.

The method (14) began by extracting the 7x7 window of pixels around the peak of the autocorrelation plane. Each pixel was then normalized by dividing it by the square root of the sum of the squares of each point in the 7x7 window. The data was considered to be a vector in 49 dimensional space, and Lt. Troxel then used a distance measurement to try to determine the identity of the target. Based on this method, a 7x7 window was extracted around each peak in the correlation of the template with search scene. Each window was normalized in the manner described, and the Euclidean distance between that peak and the peak of the template autocorrelation was measured. Euclidean distance was based on

$$D = \left(\sum_{i=1}^{49} (x_{1i} - x_{2i})^2 \right)^{\frac{1}{2}} \tag{3}$$

and the peak with the smallest Euclidean distance from the template autocorrelation was considered to be the location of the designated target.

This method was tested, but did not perform correctly. At some point in each sequence for every target, the system still switched from the target of interest to one of the other targets. The reason for this was that the normalization process removed target brightness information, and therefore, the comparison was being based only on the shape of the object in the search scene. Because there were some changes in size, shape, or orientation in the designated target from frame to frame, situations

were being created where the appearance of a target in the past scene looked more like another target than itself in the current scene.

To introduce the brightness of the target as a factor in peak analysis, the normalization of the 7x7 pixel array was removed from the process. However, when the unmodified 7x7 windows were used for comparison, error were still encountered in the tracking process. These errors did not shift to other targets often, but instead were shifting to areas in the background pixels. This was a step in the proper direction because the system was distinguishing between targets, but the system still needed refining. Areas in the noise that were relatively flat were encountering small distances between the outer pixels of the 7x7 window, but were not matching well near the peak. But the match at the outer pixels was outweighing the mismatch near the peak and these areas were yielding smaller distances than the actual peak of interest. A conceptual, 2-dimensional example of what was encountered is shown in Figure 8. In this figure, (a) represents the template autocorrelation, (b) represents the matching of (a) with the template-target correlation, and (c) represents the matching of (a) with template-noise correlation. It can be seen from Figure 8 that even though the template-target correlation matched the shape of the template autocorrelation better than the template-noise correlation, the template-noise correlation would yield a smaller distance from the template autocorrelation than would the template-target correlation. This would cause the tracker to begin tracking on background noise.

The next step was to place weighting factors on the distances between individual pixels so that matching near the peak was more important than matching near the edges of the 7x7 window. This yielded the weighted Euclidean distance metric

$$D = \left(\sum_{i=1}^{49} w_i \left(x_{1i} - x_{2i} \right)^2 \right)^{\frac{1}{2}} \tag{4}$$

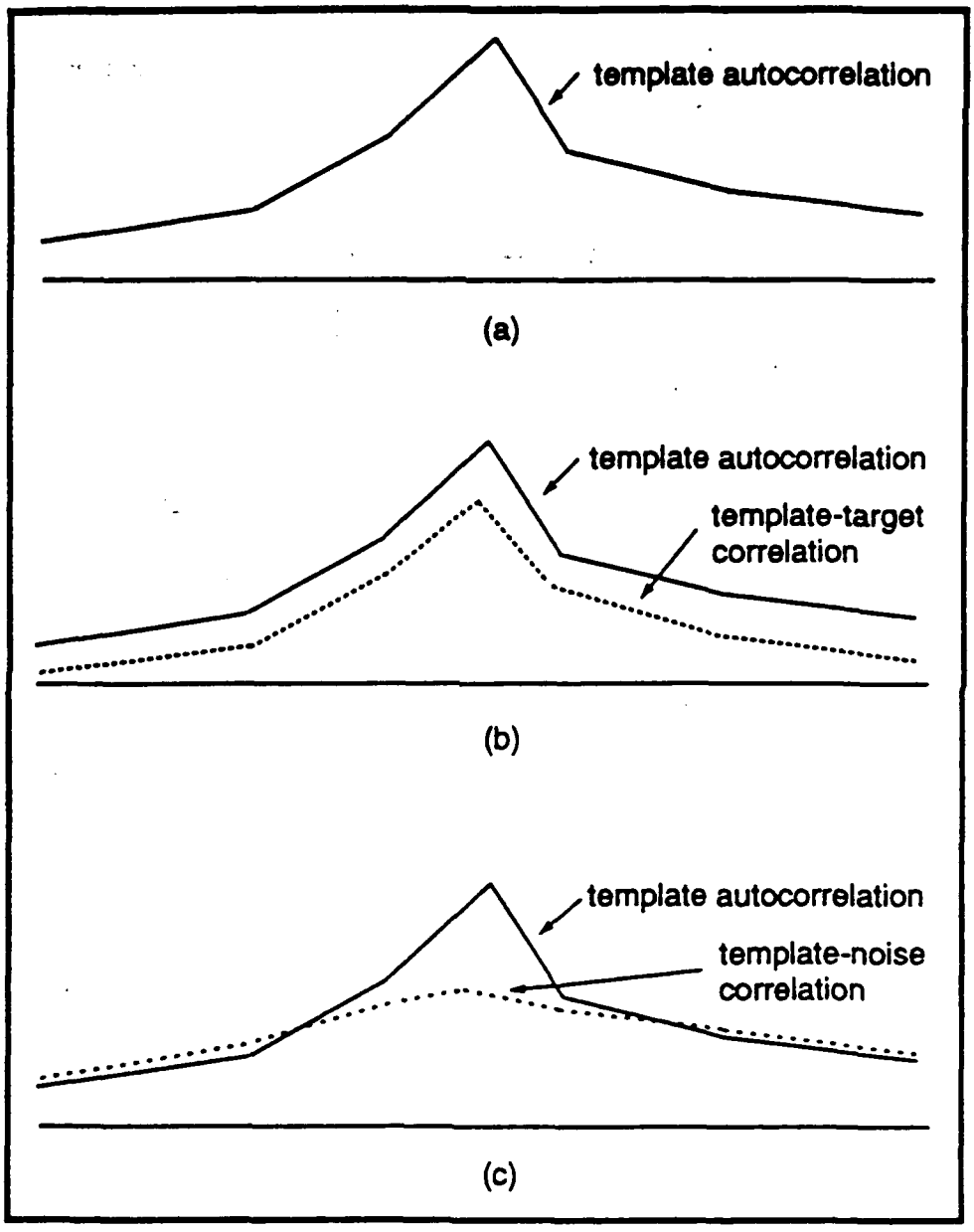


Figure 8. Correlation peak matching. (a) template autocorrelation, (b) comparison with template-target correlation, (c) comparison with template-noise correlation

where w_i is the weighting factor. The weighting factors were chosen to stress the number of rows and columns of offset that each pixel in the 7x7 window represented in the correlation plane. In Figure 9 below, the total number of rows and columns of offset for each pixel in the window is shown, with the center pixel representing zero or perfect match of the data.

6	5	4	3	4	5	6
5	4	3	2	3	4	5
4	3	2	1	2	3	4
3	2	1	0	1	2	3
4	3	2	1	2	3	4
5	4	3	2	3	4	5
6	5	4	3	4	5	6

Figure 9. Correlation offset versus pixel location.

Based on this, it was decided that pixels representing equal amounts of offset should be weighted equally. It was also chosen that each step closer to zero offset should be weighted twice the amount of one more offset. This yielded the set of weight factors in Figure 10. This set of weighting factors was tested, and still errors existed.

Analysis of the points which caused errors showed that the most important pixel in identifying the proper peak should be the height of the peak or center pixel itself. However, a look at the above weighting factors shows that the outer ring, representing the most offset, has a combined weighting of 84. The next ring has a combined weighting of 144, the third ring has a weight 192, and the peak has only a weighting of 64. Therefore, although the center pixel had the greatest weighting, it was still not as important to the total distance as the outermost ring. This created

1	2	4	8	4	2	1
2	4	8	16	8	4	2
4	8	16	32	16	8	4
8	16	32	64	32	16	8
4	8	16	32	16	8	4
2	4	8	16	8	4	2
1	2	4	8	4	2	1

Figure 10. Pixel weight factors, Set 1.

the problem that the height of the peak was still not as important as the shape of the peak.

Finally, a weighting method which emphasized the height as much as the shape was introduced. The idea was to make the center weight slightly greater than the combined weighting of the rest of the pixels. This yielded the weighting system in Figure 11.

1	2	4	8	4	2	1
2	4	8	16	8	4	2
4	8	16	32	16	8	4
8	16	32	448	32	16	8
4	8	16	32	16	8	4
2	4	8	16	8	4	2
1	2	4	8	4	2	1

Figure 11. Pixel weight factors, Set 2.

As will be shown in Chapter 4, this weighting system was quite effective. Some errors were still encountered which leaves open the reevaluation of the weighting factors for future research. Most of the errors were due to the fact that no point

had a very good match to the template autocorrelation peak. Because of this, the first fallback measure was introduced. When the distances measured for good tracks were evaluated, they all seemed to fall under certain value. Therefore, this value was multiplied by 10, and if no distances were measured less than that value, then the program assumed it had not found the target, and chose the targets last known location as its current location. This compensated for all of the errors encountered that were not corrected by the final fallback measure discussed in the next section. This allowed the system to continue tracking the proper target in all cases.

To validate this concept of a correlation postprocessing algorithm, several images not from the two sequences were chosen to see if the algorithm could find the designated target. In each case, the location of the target, as well as the box encompassing the target, were given. The box encompassing the target was used as the true template, and all template autocorrelations and template-scene correlations were performed within Khoros (9), and the correlation plane information was stored in files. The postprocessing algorithm was then implemented in a small C language program called `COR.C` (see Appendix B.3), which read in the correlation files and processed them to determine the location of the target.

The images used to validate this algorithm, along with the templates extracted, are shown in Figures 12, 13, and 14. These images include airplanes, trucks, and tanks as the target of interest. In the images which have airplanes or trucks, the target were brighter than the background, and thus a peak was expected at their location in the correlation plane. In the case of the images which had a tank as the target, the tank was not brighter than the background. Therefore, no peak in the correlation plane was expected. However, the height and shape of the template autocorrelation still could be used to try and locate the target within the scene. In all cases, the correlation plane postprocessing algorithm determined the exact location of the designated target.

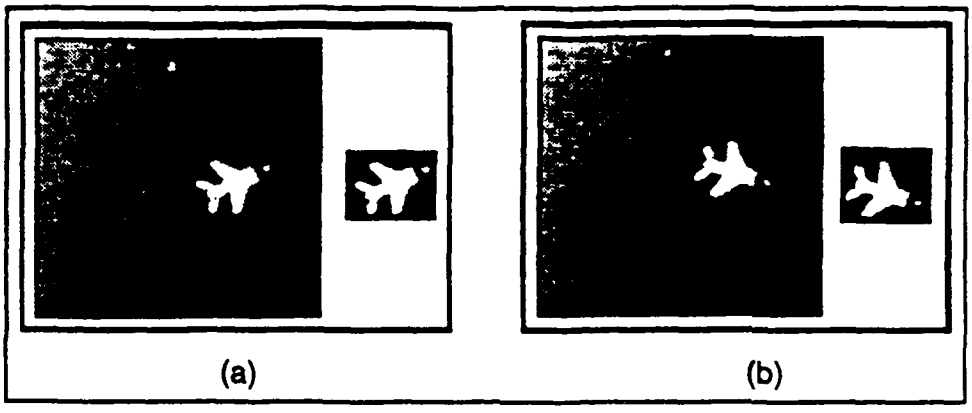


Figure 12. Images and associated templates containing an airplane used to validate the correlation plane postprocessing algorithm.

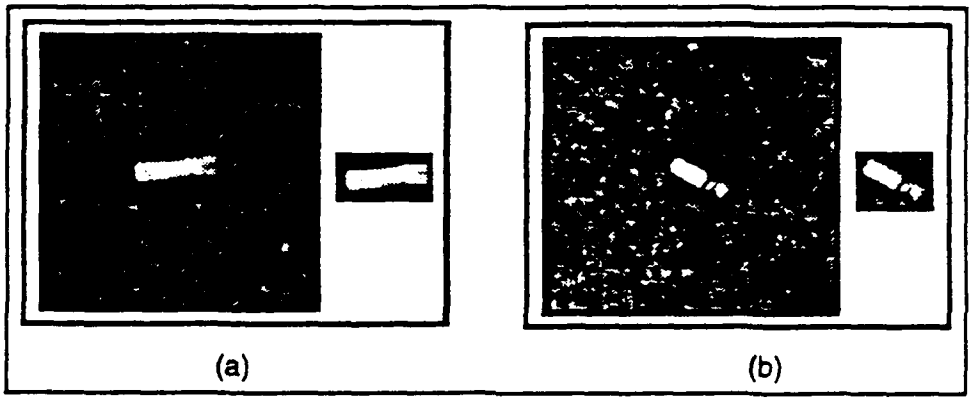


Figure 13. Images and associated templates containing a truck used to validate the correlation plane postprocessing algorithm.

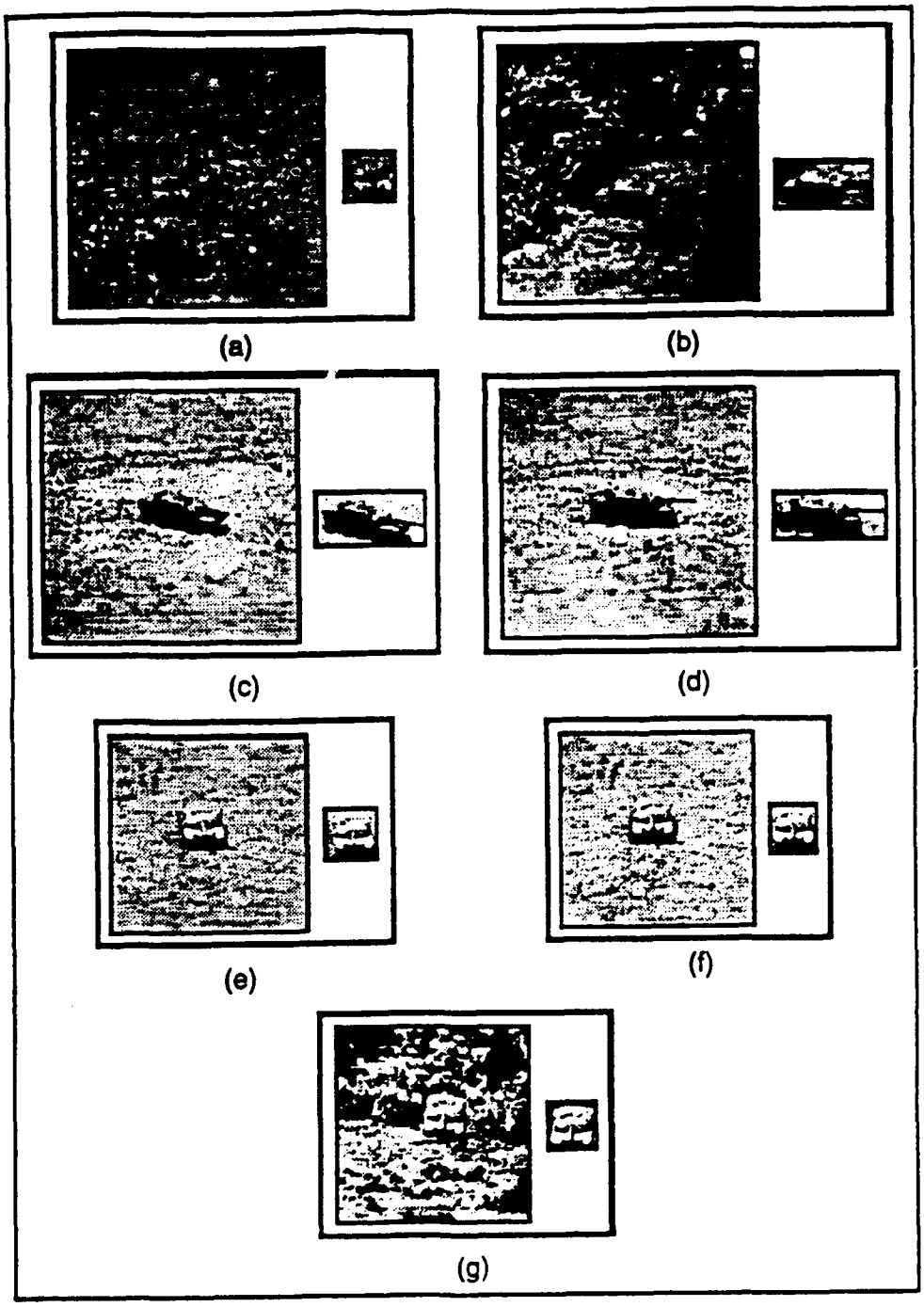


Figure 14. Images and associated templates containing a tank used to validate the correlation plane postprocessing algorithm.

3.7 Tracking Other Targets

An area of particular interest in this research was the ability to properly locate and track the locations of other targets in the search scene. The motivation for this ability was for the system to be capable of identifying, through pattern recognition, the target being tracked plus all other potential targets within the sensor's FOV. The system could then distinguish between high and low priority targets, and eventually change track to the highest priority target found. Although the ability to identify other targets and change track to a higher priority target was not investigated in this research, the ability to locate other targets in the search scene was investigated. Because the method of pattern recognition developed by Lt. Troxel (14) was very effective in identifying shapes without reference to object brightness, it was utilized as a means of identifying other potential targets in the search scene.

The search for other targets was conducted within the algorithm at the same time as the search for the actual target of interest. As described earlier, a 7x7 window of correlation information was extracted around each peak, the pixel values were normalized, and the distance from the 7x7 window of information generated by the autocorrelation of the template was determined. The algorithm stored the locations of the ten best matches within the search scene. These locations were then compared to the 10 best matched locations found in the previous scene. If any location from the search scene was within 20 pixels (a large amount of movement from frame to frame) of a stored location from the previous scene, the algorithm considered that location to represent a potential target. These locations were stored in a file by the program, and were later compared to the known locations of the other targets in the sequences to determine if the method of target location identification was successful.

Based on the relative success of this algorithm, the final fallback measure was introduced. Once the program had located the target and the ten potential target locations in the search scene, a comparison was conducted to determine if any of the

potential target locations were closer to the last known location of the target being tracked than the location as determined by the tracker. Because the time between frames did not allow for significant changes in object position, a potential target located closer to the last known location of the target was assumed to be the real target of interest. This fallback measure was effective in removing most of the errors encountered by the adaptive correlation tracker.

3.8 Correlation Information

During the execution of C language programs, several important pieces of information were stored in files. In total, four files of information were created during each run of the program. One file contained the range and locations of other potential targets as found in the method described in the previous section. Another file was generated containing range and pixel error information only. This file proved valuable in generating the plots found in Chapter IV of this thesis. The third file contained range versus target location data.

The fourth file created was the main information file. This file contained the range of each frame, the location of the target as determined by the adaptive correlation tracker, and the actual location of target as determined by the MBV Laboratory. Also included in this file was the pixel error as determined by

$$Error = \sqrt{(x_{peak} - x_{true})^2 + (y_{peak} - y_{true})^2} \tag{5}$$

and the error direction. Error direction was given in degrees and was measured from the horizontal. Values ranged from 0 to 180 degrees, and 0 to -180 degrees. Finally, the distance between the template and scene correlation and the template autocorrelation was saved. This value was normalized by the height of the template autocorrelation peak to allow for smaller values and to allow comparisons from frame to frame. In the cases where correlation peak location only was used to determine

target location, this measurement was saved with a value of zero. This measurement was also used to indicate when the program had relied on a fallback measure to maintain proper track. In the case where a location identified as a potential target was selected as the true target, this measurement was given a value of 50. And in the case of no true target being found, this measurement was given a value 100.

3.9 Summary

This chapter has described the methodology implemented in the adaptive correlation tracking process. The hardware and software, as well as the specifics of the FLIR images utilized in this research were discussed briefly. The new algorithms and procedures adapted to the existing C language programs, along with justification for their use were presented in detail. And the execution of the programs was discussed.

Chapter IV will present the results obtained by the tracking programs, as well as an analysis of the results. Chapter V will present the conclusions reached through this research effort, and will recommend areas for further study.

IV. Results and Analysis

This chapter presents the results obtained by the adaptive correlation tracker, as well as a comparison and analysis. In particular, a comparison of image preprocessing techniques when used with the adaptive template, as well as a comparison of adaptive and set size templates is given. Also presented is the results of correlation plane postprocessing to improve tracking performance. Finally, results of the ability to properly locate other targets in the FOV are presented.

4.1 Comparison Criteria

For each tracking run generated by the adaptive correlation tracker, the data utilized for comparison was the error in pixels achieved for each frame. This value represented the distance in pixels from the true center of the target of interest to the target location as determined by the tracker. Figures 15 and 16 demonstrate how error effected tracking by showing two tracks of eight consecutive frames of the same target. In Figure 15, Sequence 1, Target 2 was tracked utilizing line-by-line averaging with grey-scale restored, small template and search scene, target location based on largest peak in the correlation plane. In Figure 16, the same target was tracked utilizing no image preprocessing, adaptive template with full scene searched, and correlation plane postprocessing to determine the location of the target. Figure 15 (a-c) show the target being tracked to within 5 pixels of true center; in (d-f), the tracker begins to drift forward of the target's true center with errors between 5-10 pixels; and finally in (g-h), the tracker loses the target with errors between 30-40 pixels. In Figure 16, (a-h) show the target being tracked within 3 pixels of true center through the whole series of frames. In all cases, tracking began on the true center of the target in the first frame so that no initial bias was introduced. Therefore, all tracking errors encountered could be attributed directly to the tracking method utilized. This was done to make a comparison between basing target location on the

correlation peak with set template and scene sizes, and finding the proper correlation peak through postprocessing in conjunction with an adaptive template and full scene search. As can be seen in Figure 17, the method which utilized set template and scene sizes, along with basing target location on the location of the highest correlation peak was very sensitive to initial error offset when tracking, while the method which utilized an adaptive template and correlation plane postprocessing to find the proper peak in the correlation plane which represented the designated target was extremely insensitive to an initial error bias.

4.2 Thresholding Technique Comparison

To determine which thresholding technique worked best with the adaptive template, all three targets of Sequence 1 were tracked using the adaptive template with the small search scene option. This meant that target location was based only on the location of the tallest peak in the correlation plane. The thresholding techniques compared were scene averaging with grey-scale restored, Cline averaging with grey scale restored, and line-by-line averaging with grey-scale restored (see Section 3.3 for details). The option to restore grey-scale in all cases was used because Capt. Law (10) had found this worked best for the correlation based tracker. In Figure 18, (a) represents target 1, (b) target 2, and (c) target 3. For each target, the tracks by all three thresholding techniques are shown. Even relatively small pixel errors can effect the ability of the tracker to remain on target. In Figure 19, an error of approximately eight pixels in (a) still remains on the target, while an error of approximately twelve pixels in (b) shows that the tracker clearly has left the target. The average error versus thresholding technique for each target is shown in Table 1. Based on the average errors encountered, along with the error for each target at the final frame, scene averaging with grey-scale restored was chosen as the best binarization technique to use with the adaptive template. It was therefore the only binarization technique compared with no image preprocessing, and no binarization

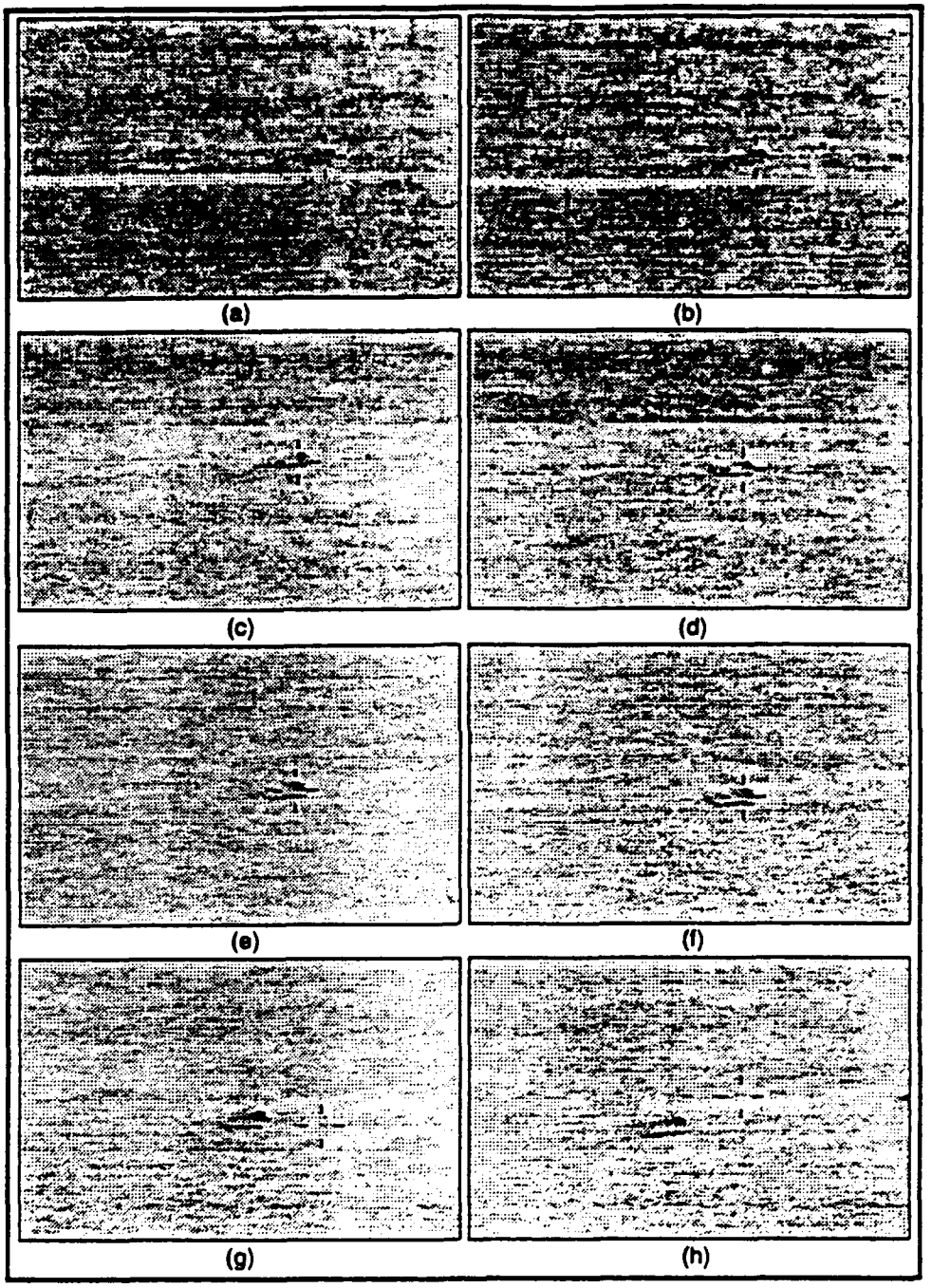


Figure 15. Tracker losing Sequence 1, Target 2 while utilizing line-by-line averaging with grey-scale restored, small template and search scene, target location based on largest peak in correlation plane. Track shows eight consecutive frames. (a-c) target is tracked within 3 pixels of true center, (d-f) error increases to between 5-10 pixels of true center, (g-h) algorithm loses track of target, errors between 30-40 pixels.

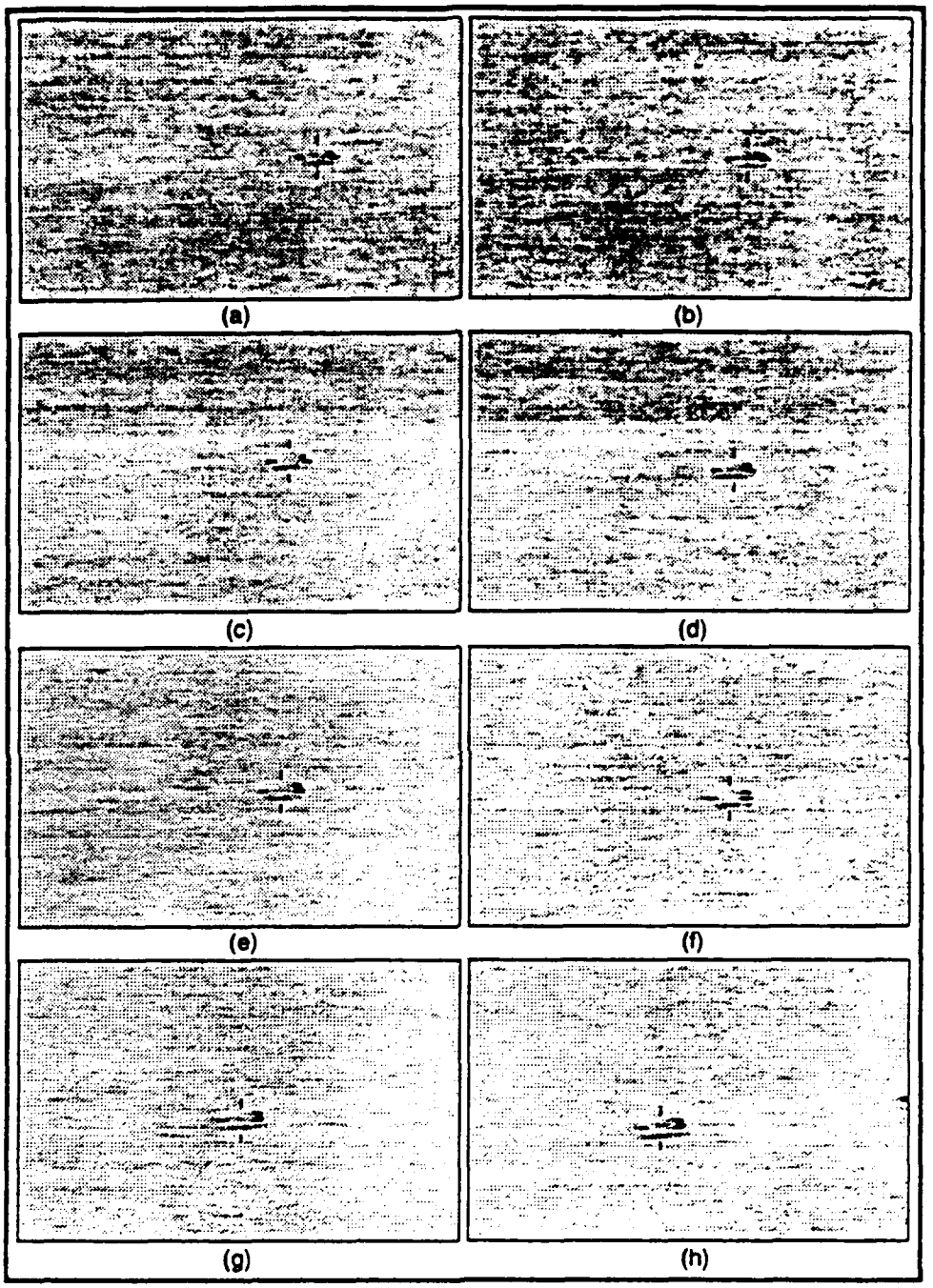


Figure 16. Good track of Sequence 1, Target 2, utilizing no image preprocessing, adaptive template with full scene searched, correlation plane postprocessing to determine target location. Track shows eight consecutive frames. (a-h) tracks target to within 3 pixels of true target center for all frames.

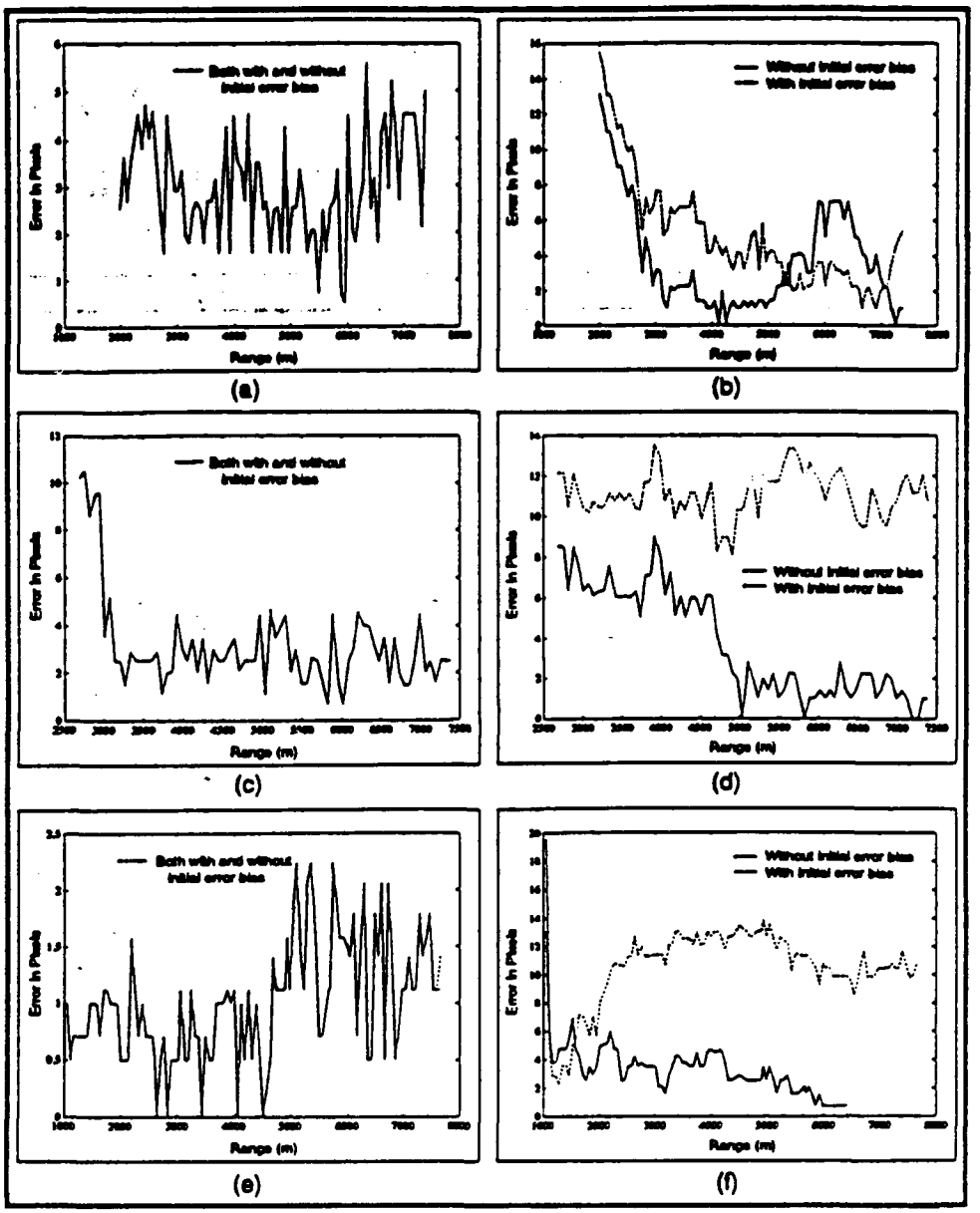


Figure 17. Effects of introducing an error in the initial designation of the target. (a,c,e) no image preprocessing, adaptive template with full scene search, correlation plane postprocessing, (b,d,f) line-by-line binarization with grey-scale restored, small template and search scene, target location based on highest correlation peak. (a,b) Sequence 1, Target 1, initial bias of 4.47 pixels (c,d) Sequence 1, Target 3, initial bias of 11.18 pixels (e,f) Sequence 2, Target 1, initial bias of 11.66 pixels.

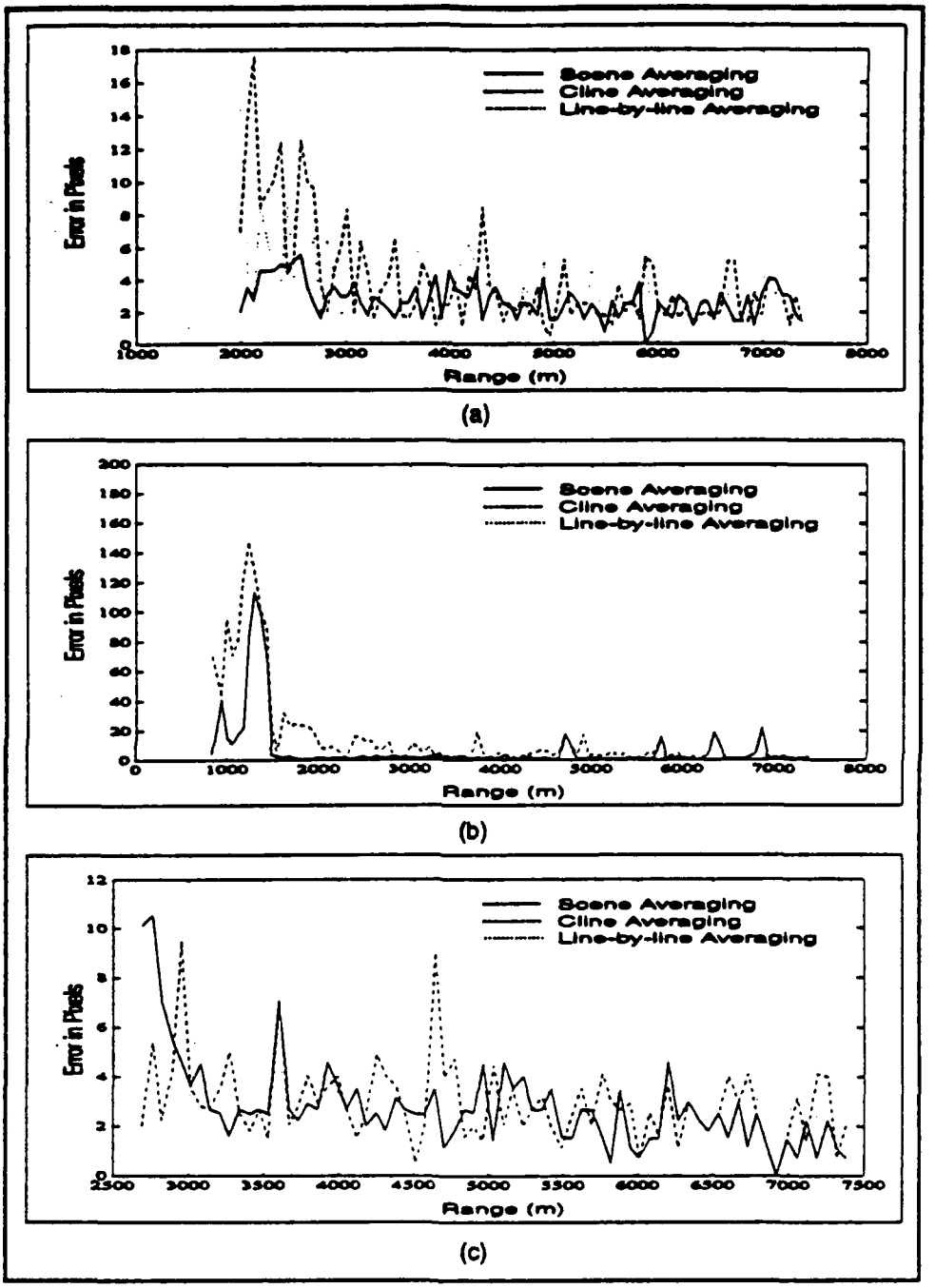


Figure 18. Comparison of thresholding techniques in range versus error for Sequence 1 (a) target 1, (b) target 2, and (c) target 3. All tracks accomplished with adaptive template, full scene searched, and correlation plane postprocessing to determine target location.

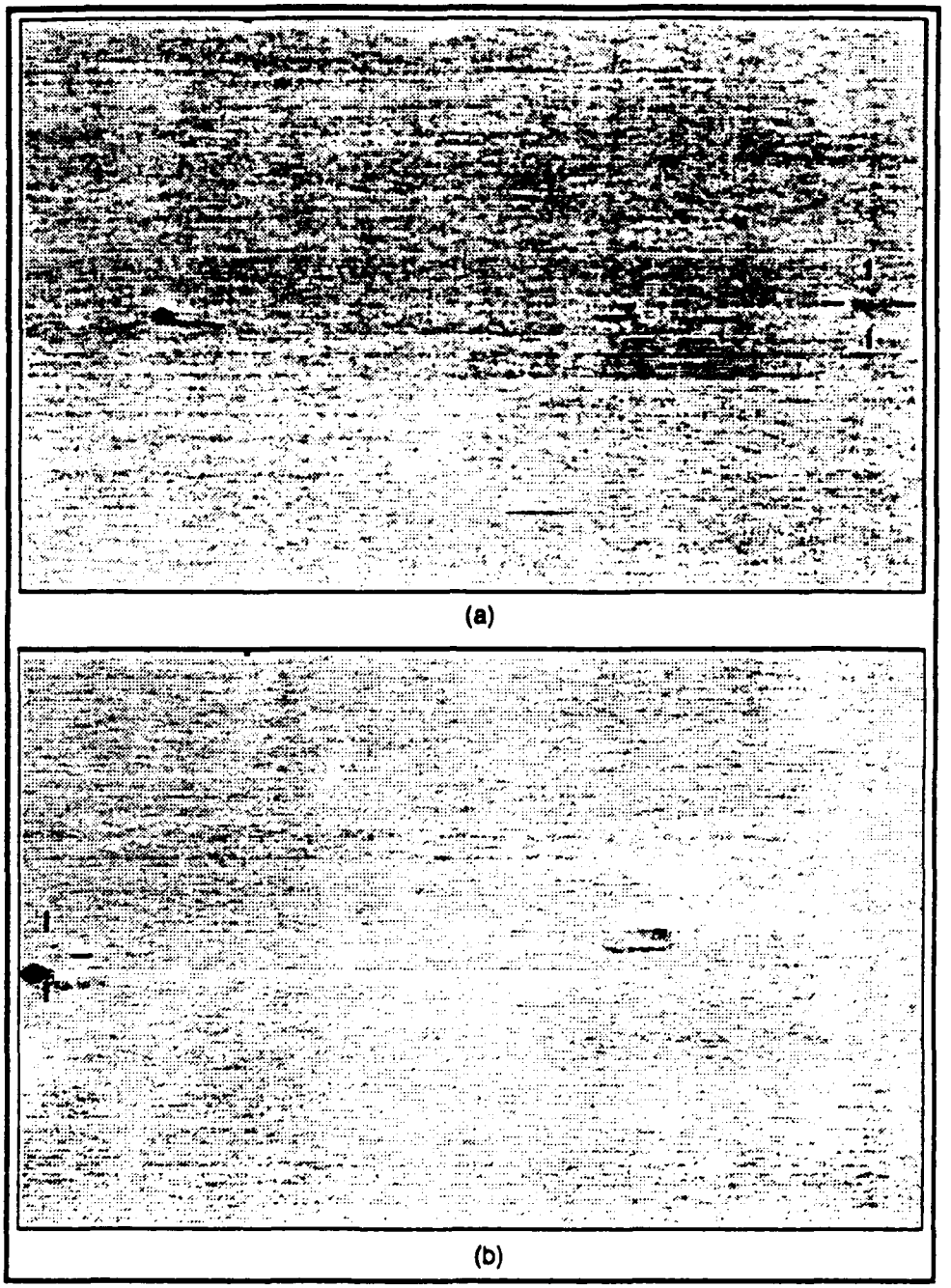


Figure 19. Effect of error on tracking ability. (a) an error of approximately eight pixels remains on target, (b) while tracking another target, an error of approximately twelve pixels is clearly off target.

with edge extraction in this study. It was chosen because it had the lowest average errors, along with the smallest average error in the final frame. This differed from the results obtained by Capt. Law (10) when a set template size was used. For a set template, line-by-line averaging with grey-scale restored was judged to yield the best results.

Table 1. Average tracking error in pixels versus thresholding technique for each target of Sequence 1 (see Figure 2). Average based on average error over entire track from approximately 7.5 km distance to terminal.

Thresholding Technique	Average Error in Pixels		
	Target 1	Target 2	Target 3
Scene Averaging w/grey-scale restored	2.67	6.22	2.81
Cline Averaging w/grey-scale restored	3.89	12.67	2.96
Line-by-line Avg w/grey-scale restored	3.42	7.80	2.96

4.3 Tracking in the Presence of Multiple Targets

When using the location of the tallest peak in the correlation plane for tracking in a sequence where multiple targets are present and close together, the tracking system often changed track to the brightest target in the sensors FOV. As was stated in Section 3.2, there is no error data available for Target 3 closer than 2000 meters. For this data, all three targets of Sequence 2 were tracked using the adaptive template with small search scene option, and scene averaging with grey-scale restored image preprocessing. In the case of Target 1, the track had only small errors, but for Targets 2 and 3, the errors began much higher and remained higher. This can be seen in Figure 20. This occurred because the tracker immediately began drifting to Target 1, the brightest of the three, starting in the first frame no matter which target was designated. This is shown in Figure 21. Essentially, this was the same result as would be expected if a hot-spot tracker had been used. Since Target 1 in both

sequences was much brighter than the other two, any hot-spot tracker utilizing the full scene as a FOV would track Target 1 no matter which target had been originally designated.

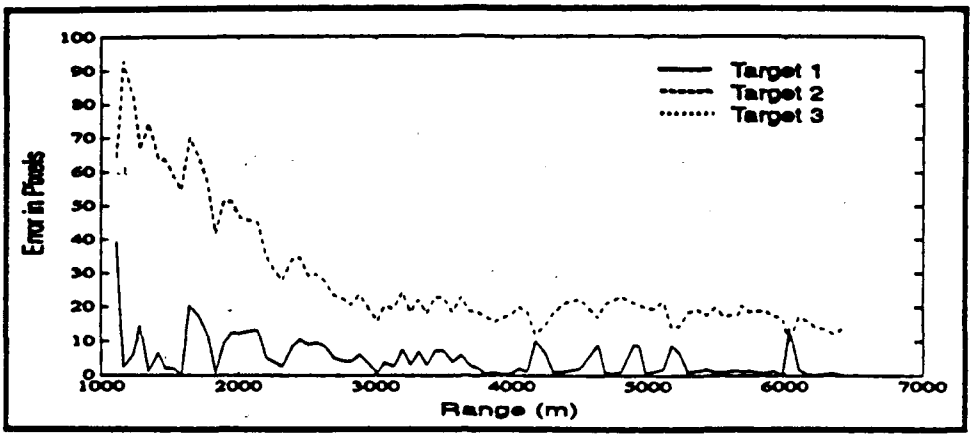


Figure 20. Tracking of targets in Sequence 2 without correlation postprocessing. Tracking of Target 3 ended at 2000 meters due to lack of true target location information at closer ranges.

To demonstrate the ability of the correlation plane postprocessing technique to track a target in the presence of multiple targets, tracking runs on all three targets in each sequence were accomplished. These runs utilized the adaptive template and searched the entire frame for the proper target. Runs on each target were accomplished using scene averaging with grey-scale restored, edge extraction, and no preprocessing of the original images. Error versus range for each run is shown in Figure 22 for scene averaging with grey-scale restored, Figure 23 for edge extraction, and Figure 24 for no preprocessing. In each figure, Sequence 1 targets 1-3 are found in (a-c), and Sequence 2 targets 1-3 are found in (d-f). The symbols x and o were placed on these graphs to indicate a frame where a fallback measure to maintain track was utilized. The presence of an x indicates that the target location determined by the adaptive correlation postprocessing algorithm was not the designated target. However, as mentioned in Section 3.7, one of the locations designated as a potential

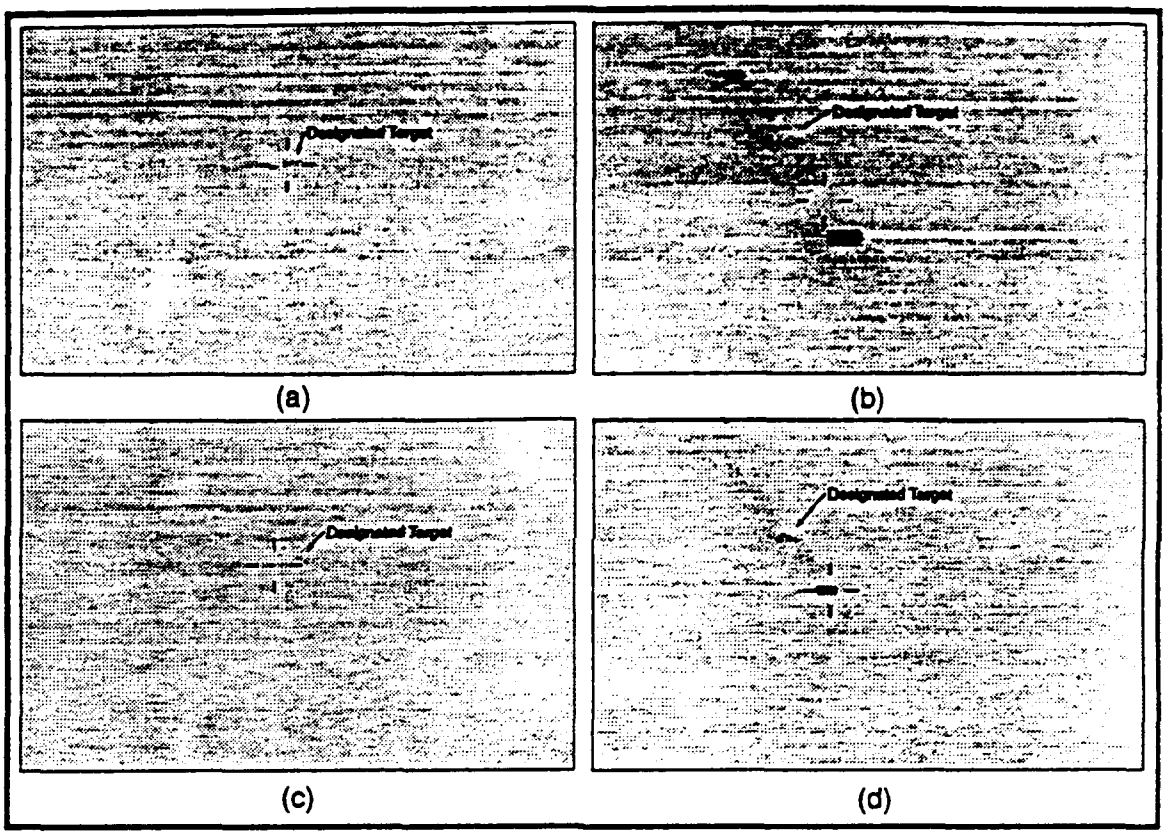


Figure 21. Drift of correlation peak tracker to Target 1 in Sequence 2 when either Target 2 or 3 was designated. (a) track point at 5980 meters when Target 2 designated, (b) track point at 1050 meters when Target 2 designated, (c) track point at 5980 meters when Target 3 designated, (d) track point at 2140 meters when Target 3 designated.

target was judged to be the proper target. The potential target was chosen as the true location of the designated target because it was closer to the last known location of the designated target in the previous scene than the location of the object chosen to be the target in the current scene. The presence of an o indicates that no correlation peak match was attained, and the algorithm considered the known location of the target in the previous scene as its new location in the current scene.

Figures 22, 23, and 24 indicate that in all three preprocessing techniques utilized the adaptive correlation tracker with correlation plane postprocessing was able to accurately track the designated target even in the presence of other, similar targets. However, a major distinguishing factor between the three preprocessing techniques was in the number of frames in which the wrong target was originally chosen and a fallback measure had to be utilized. Table 2 compares the number of frames in which the fallback measures were utilized for each image preprocessing technique. In Table 2, data is given in the form of 'x/y' where x represents the number of frames where the actual target was chosen from the pool of potential targets, and y represents the number of times that no target was found and the last known location of the target was utilized.

In the area of tracking and average tracking error, both the edge extrac ion and no preprocessing of the images performed equally well. The option of scene averaging with grey-scale restored did perform well on Sequence 1, but did not perform as well on Sequence 2. However, when the number of frames which utilized fallback measures was taken into account, the option of no image preprocessing was clearly the best of the three options. This option was able to distinguish between multiple targets in the sensor's FOV, while also maintaining very accurate tracking on the designated target. An added benefit of this option was the reduced processing time required when the frames were taken in their original form without data reduction.

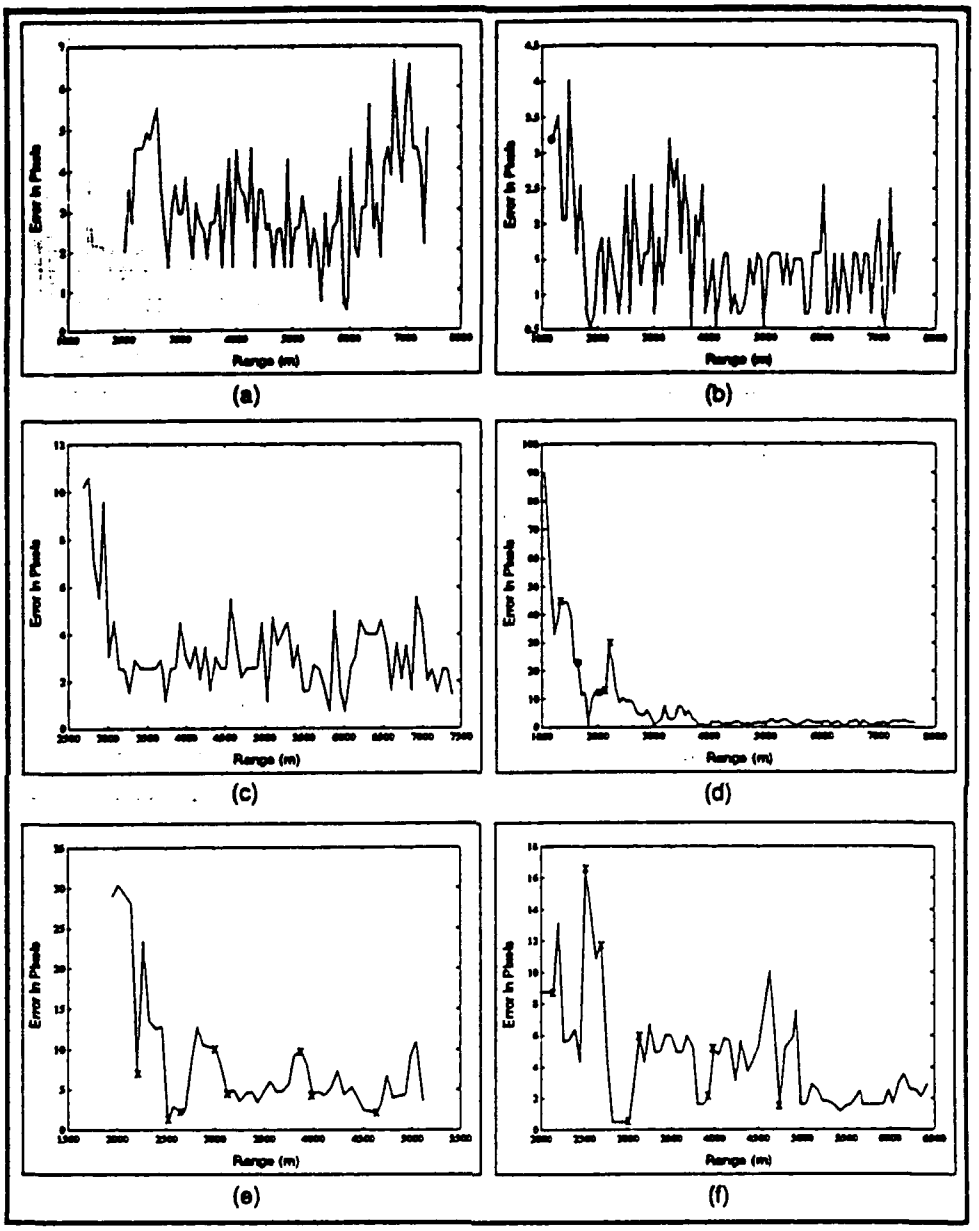


Figure 22. Range versus error tracking based on correlation plane postprocessing using scene-average w/grey-scale restored image preprocessing. Sequence 1, (a) target 1, (b) target 2, (c) target 3, and Sequence 2 (d) target 1, (e) target 2, (f) target 3. An x indicates the selected target was not originally located, but was chosen from the potential targets, while an o indicates that no target was found and the tracker used the last known location of the target.

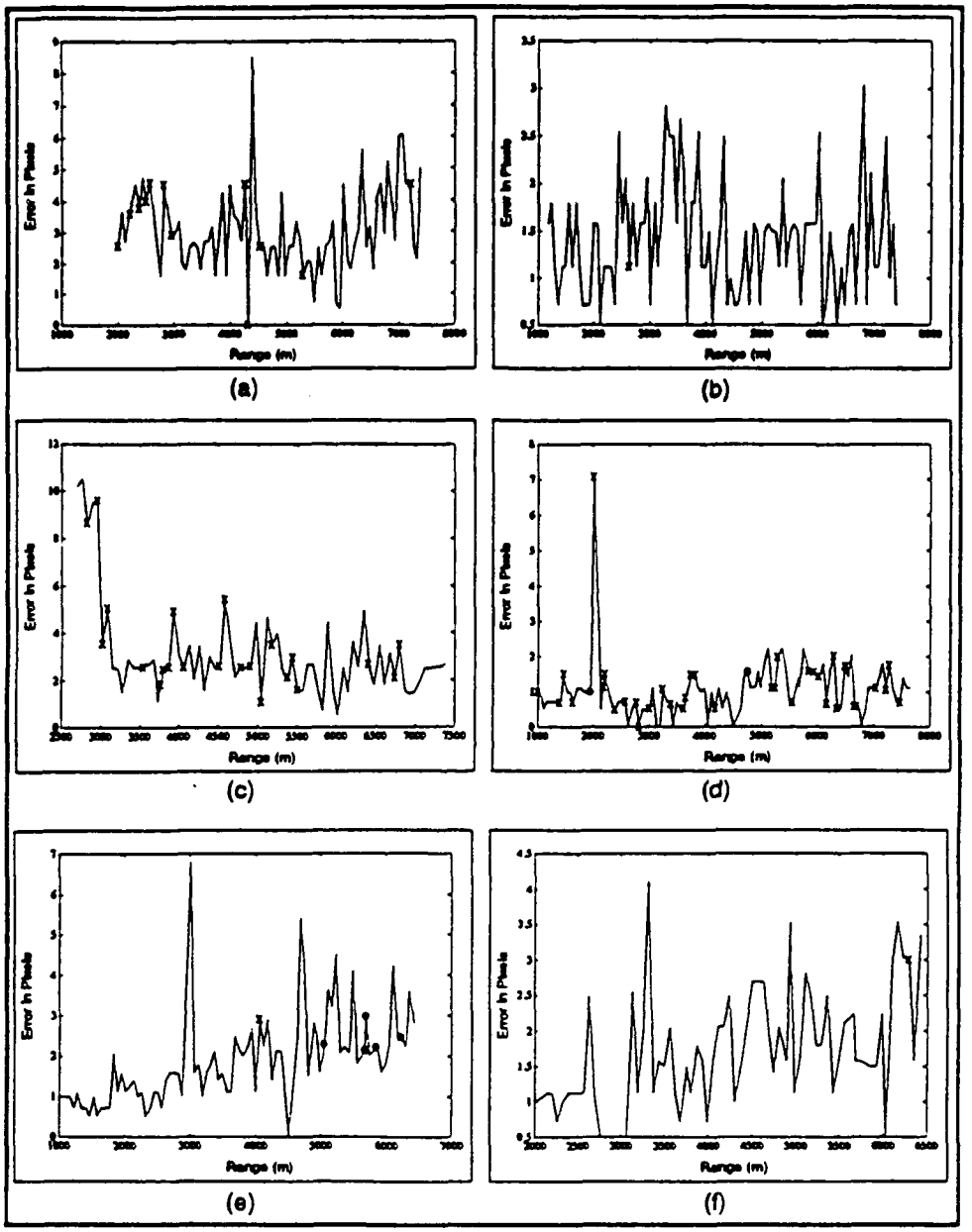


Figure 23. Range versus error tracking based on correlation plane postprocessing using edge extraction image preprocessing. Sequence 1, (a) target 1, (b) target 2, (c) target 3, and Sequence 2 (d) target 1, (e) target 2, (f) target 3. An x indicates the selected target was not originally located, but was chosen from the potential targets, while an o indicates that no target was found and the tracker used the last known location of the target.

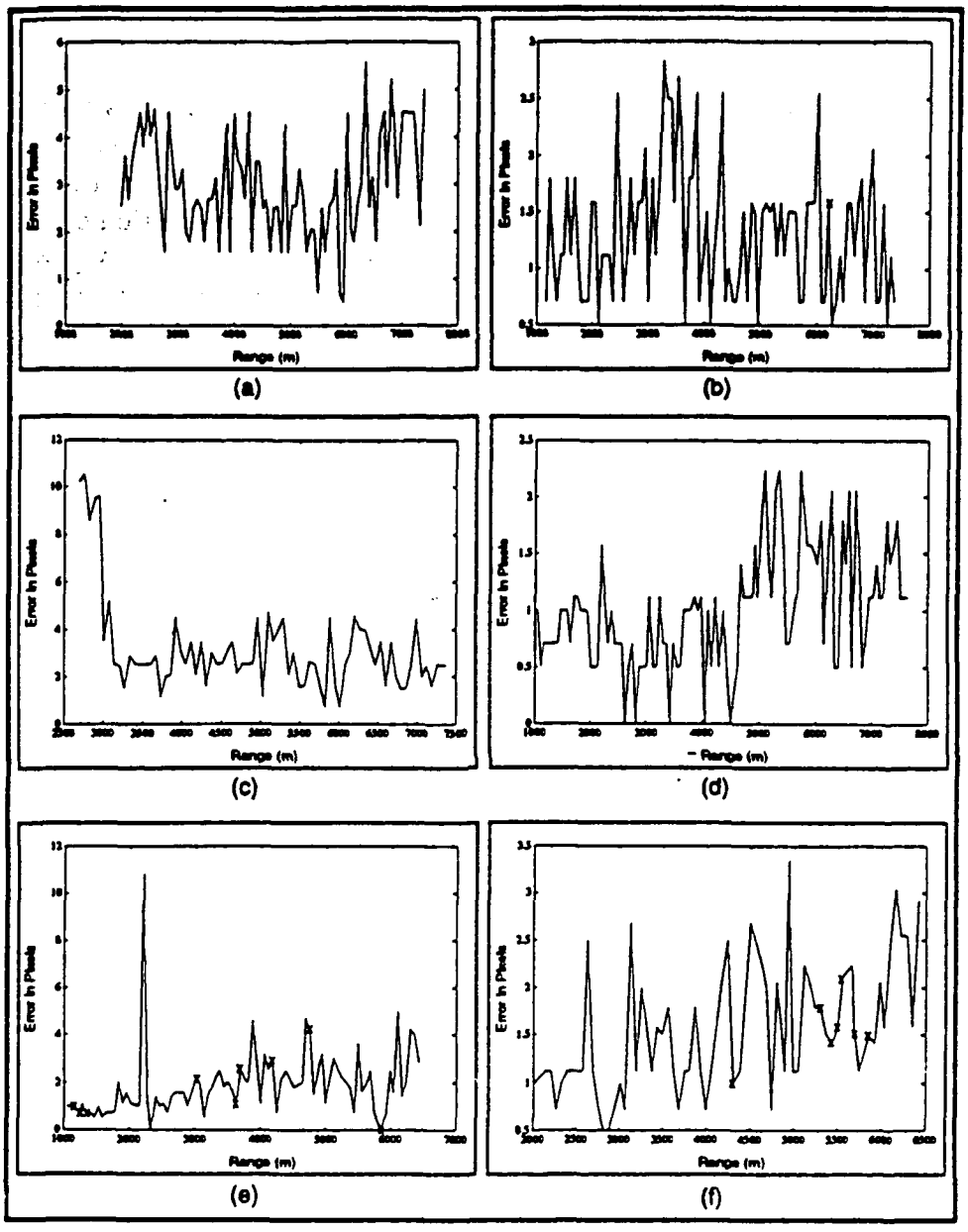


Figure 24. Range versus error tracking based on correlation plane postprocessing using no image preprocessing. Sequence 1, (a) target 1, (b) target 2, (c) target 3, and Sequence 2 (d) target 1, (e) target 2, (f) target 3. An x indicates the selected target was not originally located, but was chosen from the potential targets, while an o indicates that no target was found and the tracker used the last known location of the target.

Table 2. Comparison of number of fallback measures utilized for each image preprocessing technique with correlation plane postprocessing. Data is in the form x/y. An x represents the number of frames where the target location determined by the tracker was not the designated target, but the designated target's location was selected from the pool of potential targets based on its being closer to the last known location of the designated target than the location determined by the tracker. A y represents the number of frames where no target was found. Tar 2-1 is used to designate Sequence 2, Target 1.

Image Preprocessing	Number of Fallback Measures Utilized					
	Tar 1-1	Tar 1-2	Tar 1-3	Tar 2-1	Tar 2-2	Tar 2- 3
Scene Ave. w/grey-scale	0/0	0/1	0/0	4/1	8/0	8/0
Edge Extraction	12/0	1/0	22/0	37/2	1/5	1/0
None	0/0	1/0	0/0	0/0	9/0	7/0

4.4 Error Reduction and Walk-off

One of the main goals of this research was to reduce the error encountered by Capt. Law during his research (10), while also addressing the problem of correlator walk-off. Figure 25 compares the tracking achieved by the option implemented by Capt. Law, and the best tracking achieved during this research. For the comparison, tracking based on line-by-line averaging with grey-scale restored, a small template with small scene, and target location based on the largest peak in the correlation plane (10), was used to represent Capt. Law's research. As stated in Section 4.3, no image preprocessing, with an adaptive template, full scene search, and correlation postprocessing was chosen as the best option for this research. Table 3 compares the average error achieved by each method for each target, and Table 4 compares the final error achieved by each method for each target. For the rest of this section, the term 'LineAv/GS res' will be used to represent line-by-line averaging with grey-scale restored, small template and search scene, target location based on highest peak in the correlation plane (the best results obtained by Capt. Law), and the term

'NoPre/AdFull/Post' will be used to represent no image preprocessing, adaptive template with full search scene, and correlation postprocessing (the best results obtained during this research).

Table 3. Average tracking error in pixels for research comparison. Comparing line-by-line averaging with grey-scale restored, small scene and template, location based on largest correlation peak (LineAv/GS res) and no image preprocessing, adaptive template with full scene, and correlation postprocessing (NoPre/AdFull/Post).

Research	Average Error in Pixels					
	Targ 1-1	Targ 1-2	Targ 1-3	Targ 2-1	Targ 2-2	Targ 2-3
LineAv/GS res	3.71	9.95	3.71	3.37	22.57	30.70
NoPre/AdFull/Post	2.99	1.32	3.19	1.03	1.93	1.56

Table 4. Final error comparison of research efforts. Comparing line-by-line averaging with grey-scale restored, small scene and template, location based on largest correlation peak (SceneAv/GS res) and no image preprocessing, adaptive template with full scene, and correlation postprocessing (NoPre/AdFull/Post).

Research	Final Error in Pixels					
	Targ 1-1	Targ 1-2	Targ 1-3	Targ 2-1	Targ 2-2	Targ 2-3
LineAv/GS res	13.15	109.41	8.54	19.66	69.70	62.32
NoPre/AdFull/Post	2.55	0.71	10.20	1.00	1.00	1.00

Analysis of Table 3 verified that the average error encountered while tracking had been reduced for all three targets in both sequences. The average error improved from the Capt. Law's research to this research an average of 63%. This represented a significant improvement in tracking performance.

As mentioned earlier, correlator walk-off can be attributed to the differences in the target from frame to frame. Because this error is random in nature, its effects

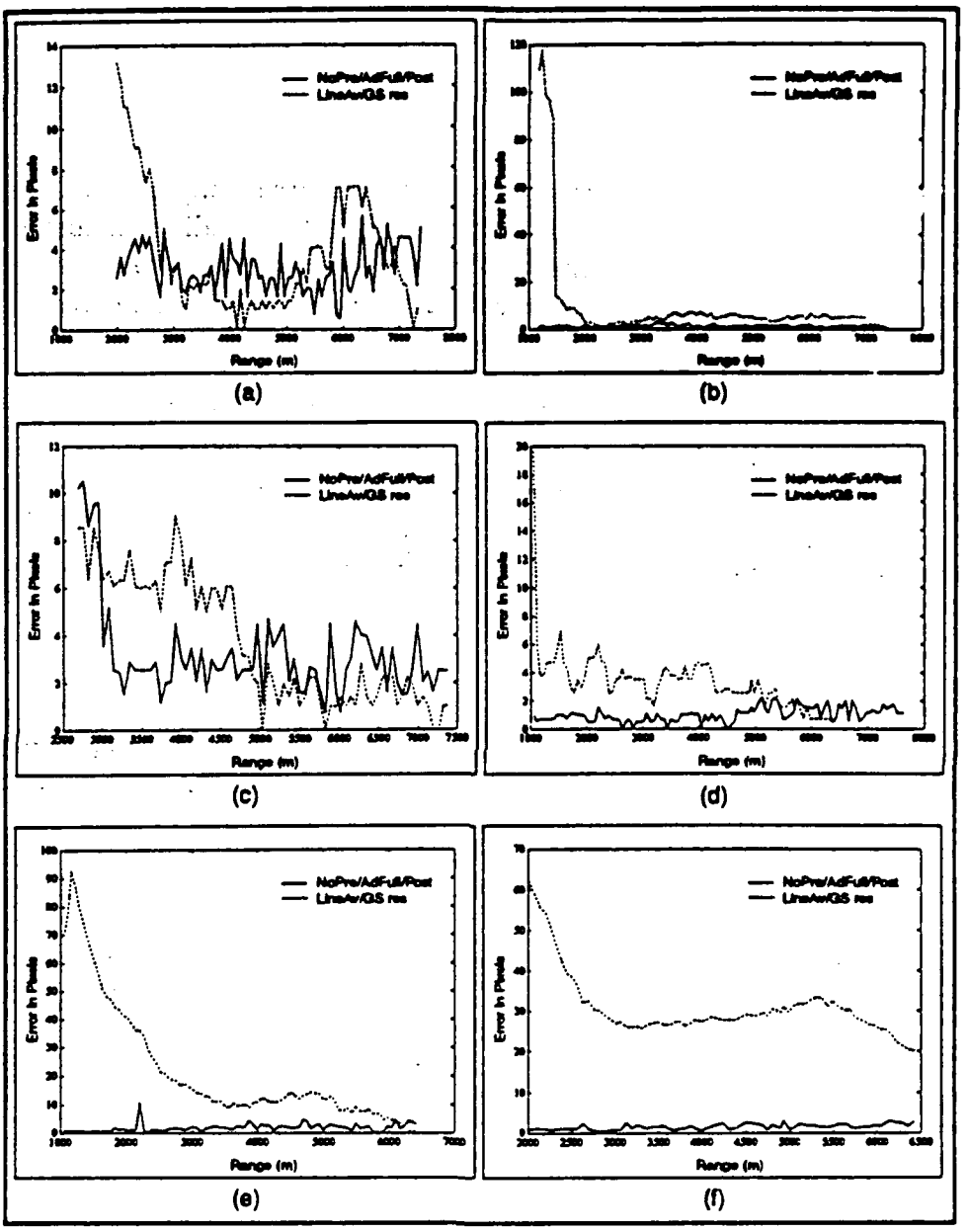


Figure 25. Comparison of best tracking results obtained by Capt. Law (10) using line-by-line averaging with grey-scale restored, small scene and template, and target location based on largest correlation peak (LineAv/GS res), and this research utilizing no image preprocessing, adaptive template with full scene, and correlation plane postprocessing (NoPre/adFull/Post). Sequence 1, (a) target 1, (b) target 2, (c) target 3, and Sequence 2 (d) target 1, (e) target 2, and (f) target 3.

are cumulative. In the situation encountered in this research, where the sensor platform was moving towards the targets, the effects of walk-off can become even more prominent as the distance between platform and target decrease because the amount of change from frame to frame becomes greater. As seen in Table 4, the error encountered in the final frame was very large, and often many times greater than the average error per frame. This was a strong indication that walk-off was having a significant effect on the tracker's performance. On the other hand, Table 4 shows that the final error encountered in the new research was very low for all but one target, and that for all but that target the final error for each target was less than the average error encountered while tracking that target. The final error encountered for Sequence 1, target 3 can be attributed to the way in which the target left the sensor's FOV. Because this exit occurred over several frames, it introduced large changes in the targets size and shape from frame to frame. This can be seen in Figure 26. However, the overall results obtained in the new research indicated that walk-off was not a significant factor in the ability of the correlation based algorithm to track.

In order to fully understand how all this data effects tracking performance, snapshots were taken of the tracking of each target utilizing both line-by-line averaging with grey-scale restored, small template and search scene, target location based on highest correlation peak, and no image preprocessing, adaptive template with full scene search, with correlation plane postprocessing to determine target location. These snapshots were taken at approximately 6000 meters range, and the final frame of each target in each sequence. The snapshots, shown in Figures 27, 28, 29, 30, 31, and 32 clearly show the difference in tracking ability between the two methods.

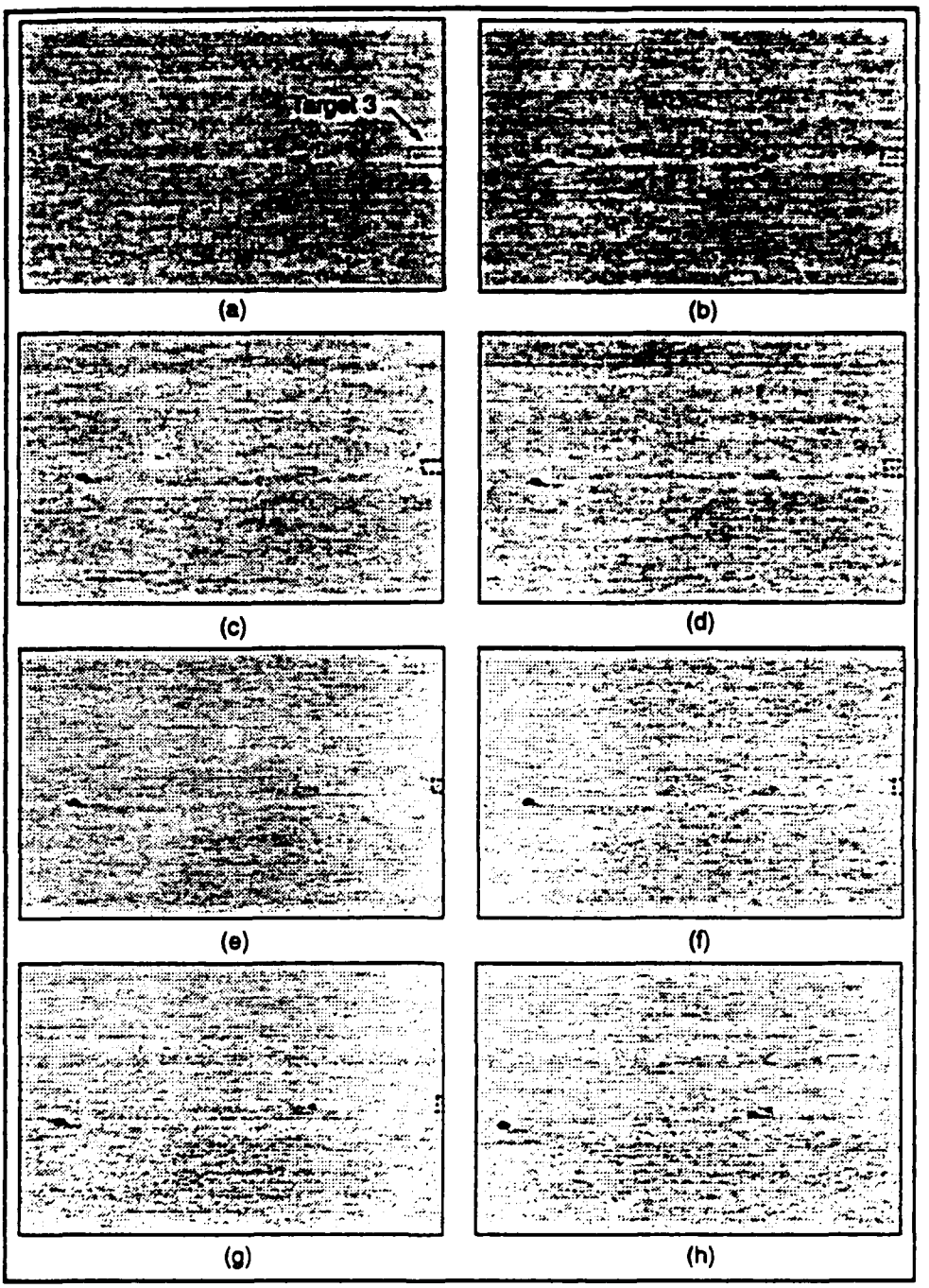


Figure 26. Frames showing the departure of Sequence 1, Target 3 from the sensor's FOV.

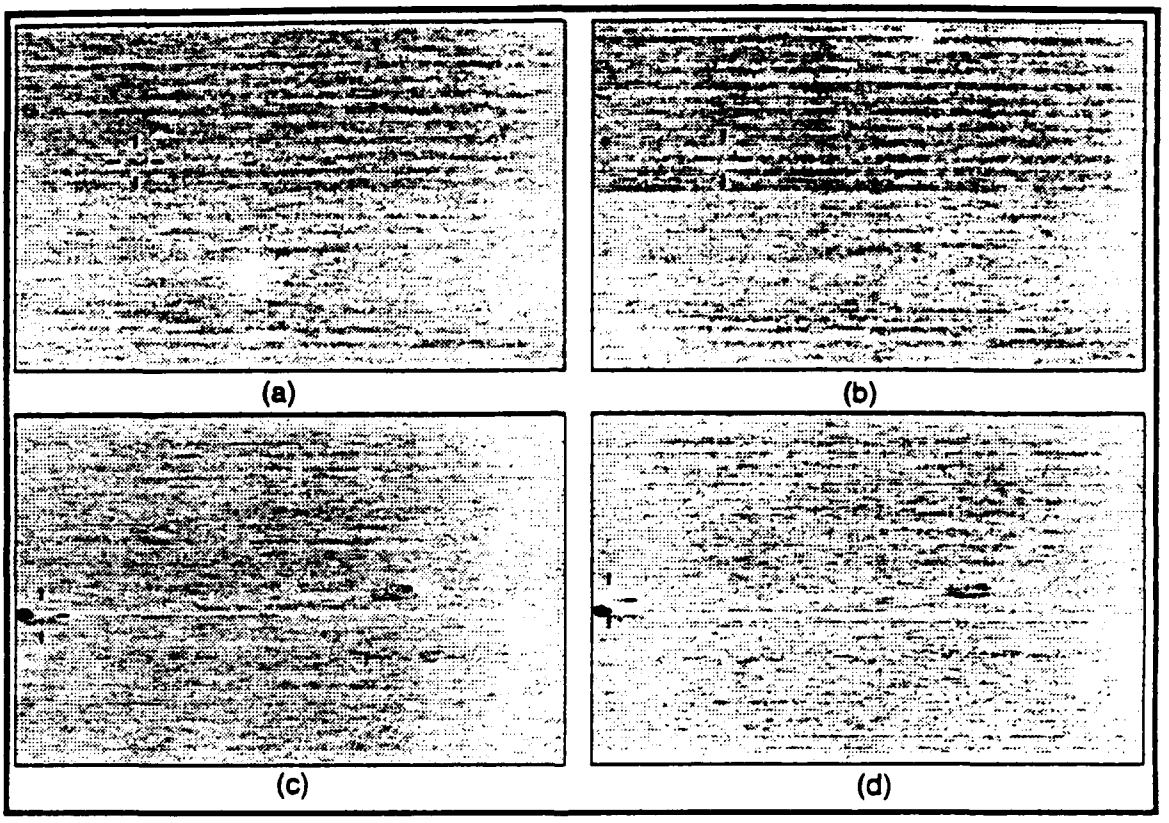


Figure 27. Snapshots of tracking Sequence 1, Target 1 using (a,c) no image preprocessing, adaptive template with full scene search, and correlation plane postprocessing to determine target location, (b,d) line-by-line binarization with grey-scale restored, small template and search scene, target location based on highest correlation peak. (a,b) taken at range of 6010 meters, (c,d) final frame of designated target.

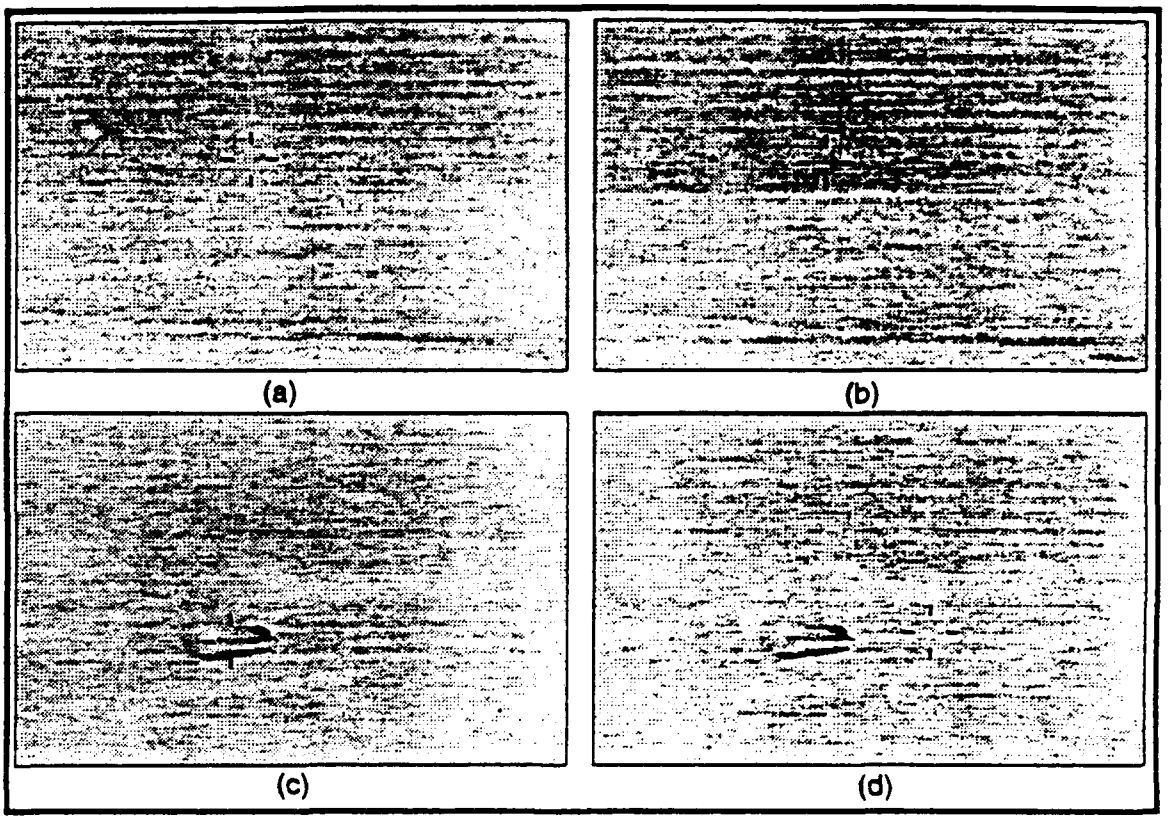


Figure 28. Snapshots of tracking Sequence 1, Target 2 using (a,c) no image preprocessing, adaptive template with full scene search, and correlation plane postprocessing to determine target location, (b,d) line-by-line binarization with grey-scale restored, small template and search scene, target location based on highest correlation peak. (a,b) taken at range of 6010 meters, (c,d) final frame of designated target.

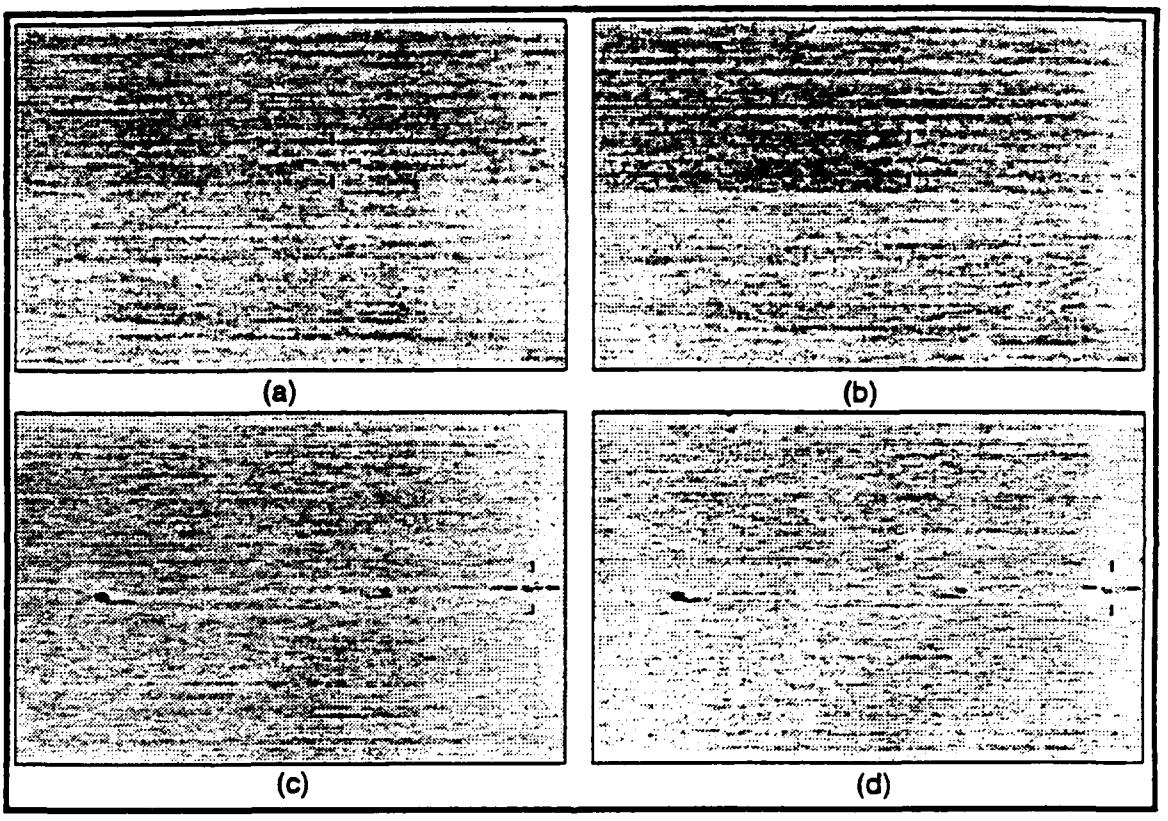


Figure 29. Snapshots of tracking Sequence 1, Target 3 using (a,c) no image preprocessing, adaptive template with full scene search, and correlation plane postprocessing to determine target location, (b,d) line-by-line binarization with grey-scale restored, small template and search scene, target location based on highest correlation peak. (a,b) taken at range of 6010 meters, (c,d) final frame of designated target.

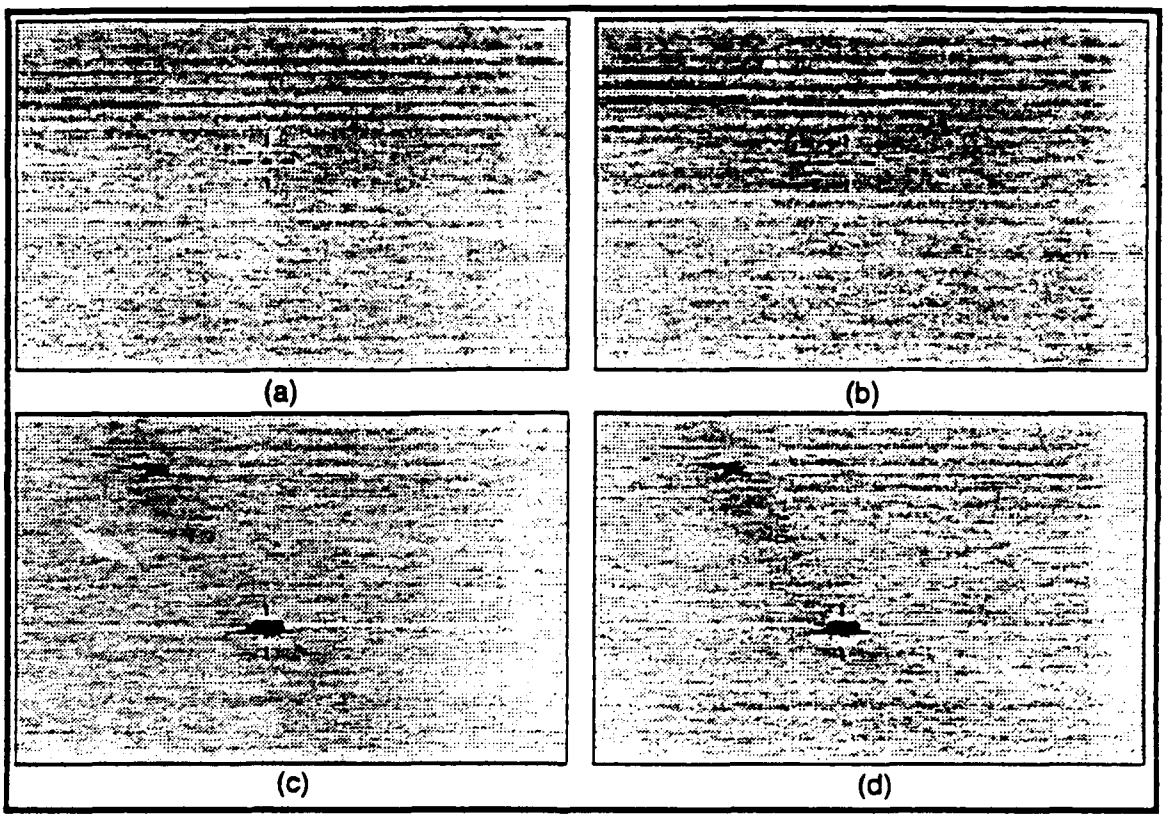


Figure 30. Snapshots of tracking Sequence 2, Target 1 using (a,c) no image preprocessing, adaptive template with full scene search, and correlation plane postprocessing to determine target location, (b,d) line-by-line binarization with grey-scale restored, small template and search scene, target location based on highest correlation peak. (a,b) taken at range of 5980 meters, (c,d) final frame of designated target.

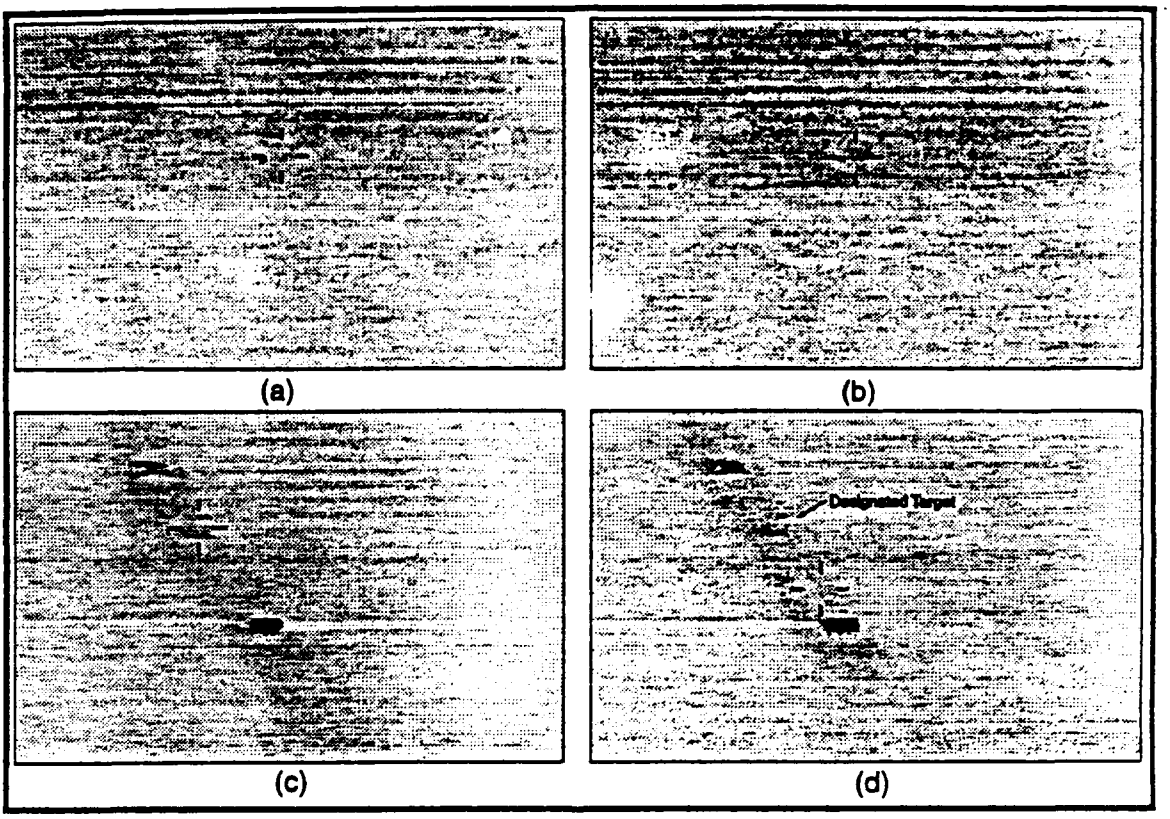


Figure 31. Snapshots of tracking Sequence 2, Target 2 using (a,c) no image preprocessing, adaptive template with full scene search, and correlation plane postprocessing to determine target location, (b,d) line-by-line binarization with grey-scale restored, small template and search scene, target location based on highest correlation peak. (a,b) taken at range of 5980 meters, (c,d) final frame of designated target.

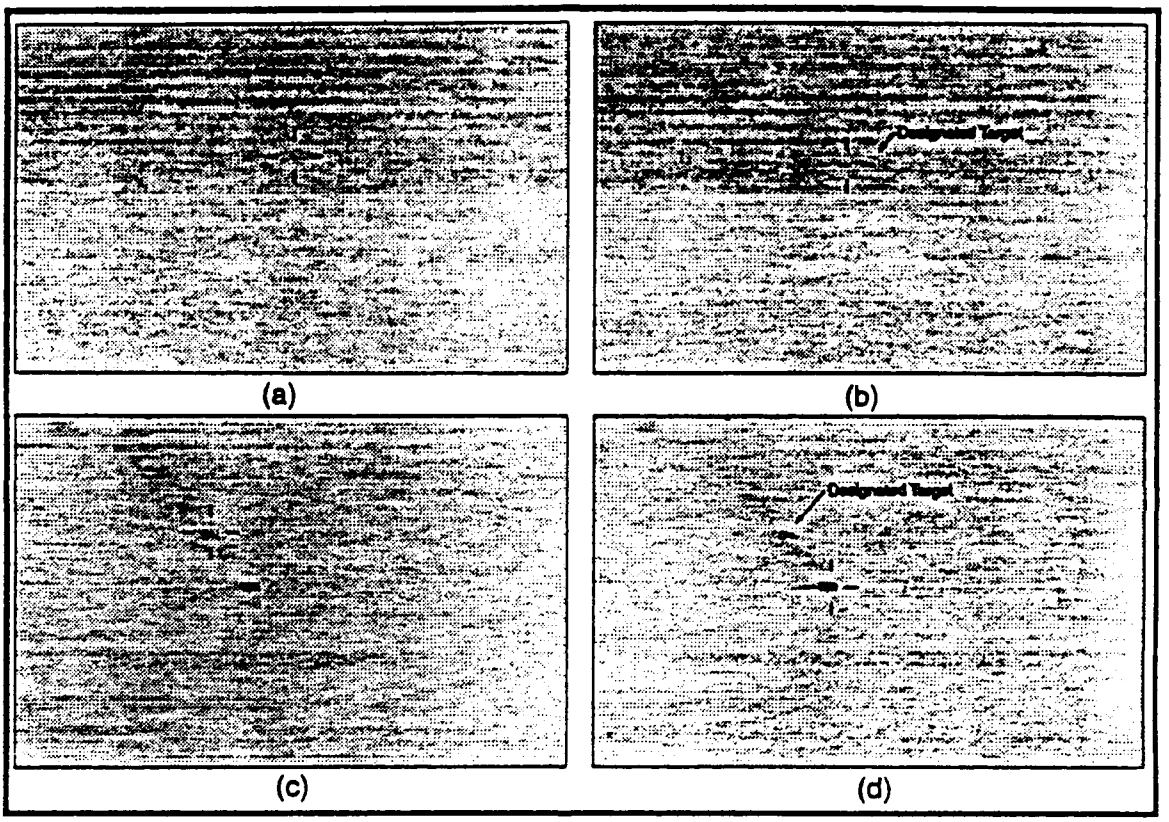


Figure 32. Snapshots of tracking Sequence 2, Target 3 using (a,c) no image preprocessing, adaptive template with full scene search, and correlation plane postprocessing to determine target location, (b,d) line-by-line binarization with grey-scale restored, small template and search scene, target location based on highest correlation peak. (a,b) taken at range of 5980 meters, (c,d) final frame of designated target.

4.5 Locating Other Targets

While the targets were being tracked using correlation plane postprocessing, the method of pattern recognition developed by Lt. Troxel (14) and discussed in Section 3.7 was utilized to try and determine the locations of other potential targets within the sensor's FOV. For a location to be considered a potential target, it must have been within a small number of pixels of a location found by this method in the previous frame. Again data was collected for three image preprocessing options. These options were scene averaging with grey-scale restored, edge extraction, and no preprocessing. For each of these, the percentage of frames in which each target was correctly located as a potential target while another target was being tracked was determined. Table 5 shows the results that were obtained when scene averaging with grey-scale restored was utilized. Table 6 contains the same data for edge extraction, and Table 7 contains the data for no image preprocessing.

Table 5. Percentage of frames where other targets were located when scene averaging with grey-scale restored was utilized.

Target Tracked	Other Targets		
	Target 1	Target 2	Target 3
Sequence 1, Target 1	-	89%	90%
Target 2	57%	-	58%
Target 3	44%	35%	-
Sequence 2, Target 1	-	35%	79%
Target 2	68%	-	37%
Target 3	68%	37%	-

Tables 5, 6, and 7 show that while this method of other target identification was not overly successful in all cases, it was able to correctly locate many of the other targets in excess of 75% of the sequence frames. As was mentioned in Section 3.7, other target location was dependent on having identified the location of that target as a potential other target in the previous frame. Therefore, this algorithm did not

Table 6. Percentage of frames where other targets were located when edge extraction was utilized.

Target Tracked	Other Targets		
	Target 1	Target 2	Target 3
Sequence 1, Target 1	-	45%	58%
Target 2	4%	-	0%
Target 3	47%	31%	-
Sequence 2, Target 1	-	18%	42%
Target 2	0%	-	0%
Target 3	2%	0%	-

Table 7. Percentage of frames where other targets were located when no image preprocessing was utilized.

Target Tracked	Other Targets		
	Target 1	Target 2	Target 3
Sequence 1, Target 1	-	77%	92%
Target 2	5%	-	58%
Target 3	0%	42%	-
Sequence 2, Target 1	-	7%	87%
Target 2	0%	-	2%
Target 3	2%	14%	-

attempt to maintain track, but only identified those frames in which the location of the other target was identified as a potential target in two consecutive frames. While some of the best results were obtained with no image preprocessing, the option which utilized scene averaging with grey-scale restored had the best overall performance. While the performance of this method was not perfect, the amount of additional processing required to accomplish it was minimal. Meanwhile, the potential benefits from knowing the locations of other targets could be tremendous. The ability of a system to identify those other targets, choose the highest priority target in its FOV, and change track to the highest value target could be extremely beneficial.

4.6 Summary

This chapter has presented the results obtained through this research effort. The ability to track a target in the presence of multiple targets utilizing correlation plane postprocessing was demonstrated. Also, the ability of this algorithm to reduce the tracking error by 63% and to eliminate the effects of correlator walk-off by reducing final frame errors by 94% and maintaining final frame errors less than the average error over the entire tracking sequence was shown. Finally, the ability to accurately determine the locations of other potential targets in the sensor's FOV was addressed. For the overall best results, the techniques which utilized no image preprocessing, an adaptive template with the full scene searched, and the correlation plane postprocessing described in Section 3.6.2 were found to be the most effective.

Chapter V will present the conclusions reached during this research and will make recommendations for potential future research efforts.

V. Conclusions

This chapter presents a brief summary of this research, provides conclusions based on the results obtained, and recommends areas for future research efforts.

5.1 Research Summary

Their were four major goals of this research: 1) to track a target in the presence of multiple, similar targets of interest; 2) to reduce the tracking error encountered by Capt. Law (10) in the previous correlation based tracker research; 3) to reduce the effects of correlator 'walk-off' in the tracking algorithm; and 4) to identify the locations of other objects of interest while tracking a designated target.

The first goal was met through the use of a correlation plane postprocessing technique. Based on the assumption that the target changed little from frame to frame, the template was autocorrelated and the 7x7 array of pixels surrounding the peak was extracted. Since the template correlated with the target in the next frame was expected to have a correlation peak of roughly the same height and shape, the template-scene correlation plane was then searched to find the peak which had the smallest distance in weighted 49-dimensional space from the template autocorrelation peak. This allowed the tracking algorithm to identify the proper peak, instead of just the largest peak, in the correlation plane which represented the location of the designated target. This postprocessing technique, in conjunction with an adaptive template which set its size to match the actual size of the target (see Section 3.5), was also used to reduce tracking error and counter the effects of correlator "walk-off." Overall, average tracking error was reduced by an average of 63% and final frame errors were reduced by an average of 94%. Finally, an algorithm which utilized the information generated by the correlation plane postprocessing method was utilized to identify the locations of other potential targets in the scene.

5.2 *Research Conclusions*

This research reinforced Capt. Law's (10) major conclusion that an adaptive correlation based tracking system can be utilized to track 'real-world' IR imaged targets as long as the following criteria was met. The criteria is that the time difference between two frames was small enough that no significant change in target size or orientation occurred. When this criterion was met, the correlation based tracking system was able to track a designated target.

The presence of multiple targets in a sensor's FOV presented a unique problem to a correlation based tracking system which identified the target based on the largest peak in the correlation plane. Eventually, any such tracking system, whether correlation based or a simple hot-spot tracker, would drift to the brightest target in the sensor's FOV. However, correlation plane postprocessing offered a means of removing this shortfall. By utilizing the 7x7 array of pixels from the template autocorrelation to find the minimum weighted distance of every peak in the template-scene correlation, the best match to the size and shape of the template autocorrelation peak in the template-scene correlation plane can be found. Using the location of the best matched peak as the designated targets location proved to be an effective measure in maintaining track on the designated target.

The combination of correlation plane postprocessing and an adaptive template algorithm proved effective in reducing the average tracking error. This research significantly reduced the tracking errors encountered by Capt. Law (10). At the same time, where Capt. Law, through testing different image preprocessing techniques and using the largest peak in the correlation plane as the target location, found it necessary to preprocess the input images in order to keep tracking error low, this research found that the combination of correlation plane postprocessing, the use of the template autocorrelation peak size and shape to find the best matched peak in the template-scene correlation plane, and an adaptive template worked best when

no image preprocessing was utilized. This represented a decrease in the overall processing time for each frame, and was therefore considered to be very desirable.

While reducing the average tracking error, the combination of correlation plane postprocessing and an adaptive template algorithm also proved effective in combating the effects of correlator walk-off. Although correlator walk-off is a random error, it impacted Capt. Law's research (10) through an increase in total error as the sensor platform closed in on the designator target. This led to large tracking errors in the final frames; errors much larger than the average tracking error of the system. However, this research yielded very small tracking errors in the final frames; errors which were smaller than the average tracking error of the system. This demonstrated that the new algorithms were able to compensate for correlator walk-off.

Finally, this research demonstrated that while utilizing correlation plane post-processing to accurately track a target in the presence of multiple targets, that the information generated can also be used to determine the locations of other targets of interest within the sensor's FOV. While the algorithm to accomplish this was not perfect, it sometimes, an average between 20% and 58% of the time and sometimes greater than 75% of the time, was effective as a means of potential target location identification.

Appendix A. *Adaptive Tracker Program Execution*

In his thesis (10), Capt. Law described the operation of the two C language programs in detail. This section provides a brief overview of the programs operation, and discusses the selection of new options implemented during this research effort.

To run the C language programs, the SUN computers must be operating the OpenWindows environment, and two separate command tools must be opened. The first of the two programs to be executed must be `'trac_menu'` (see Appendix x). Once this program is running, `'trac_prgm'` must be executed in the other command tool. The second program then asks the user whether or not images are to be displayed on the monitor. The operator who selects 'no', then has the ability to temporarily interrupt the program later when it is running and place it in the systems background to complete execution. This allows other users to operate the terminal while the programs are running.

Two other changes were made in the operation of the programs. First was the addition of the edge extractor option. After selecting an image preprocessing method, the operator was given the opportunity to select edge extraction also. The edge extraction would always occur after the initial preprocessing of the image. The second change in the programs was the addition of the two new template and search scene size options as described earlier in this chapter. It should be noted that the small, medium, large, and adaptive template with small scene options all base tracking on the highest peak in the correlation plane. Only the adaptive template with full scene option implements the correlation peak analysis algorithm and the tracking of other potential targets algorithm.

Appendix B. *C Language Computer Programs*

B.1 TRAC_MENU.C

```
/*****************************************************************/
/* TRAC_MENU.C                                                   */
/*                                                               */
/*            AIR FORCE INSTITUTE OF TECHNOLOGY                  */
/* A PROCESS TO ENABLE COMMUNICATION BETWEEN USER AND PROGRAM    */
/*                     by Capt Paul D. Law                       */
/*                       October 7, 1991                         */
/*                                                               */
/*                         modified by                           */
/*                    Capt Dennis A Montera                      */
/*                              , 1992                           */
/*                                                               */
/*                                                               */
/* This is the first of a pair of programs which must be ran     */
/* in unison in order to implement the adaptive correlation      */
/* tracking algorithm. It configures the program's required      */
/* files and variables for initial operation and provides the    */
/* menus and prompts which enable the user to communication      */
/* and control the overall adaptive correlation process.         */
/*****************************************************************/

#include <stdio.h>
#include <stdlib.h>

main()
{
  FILE *fp1, *fp2;
  int character, i, x_value, y_value, delay, step;
  double x_coord, y_coord;
  char string[10], temp1[5], temp2[5], opt = 'n', choice;
  char temp, state, response, on = '1', off = '0', makedir[64];
  char clean_up[128], EOS, it, size, binary, edger = 'n';
  float factor;

  /*-----------------------------*/
  /* Create directory under tmp. */
  /*-----------------------------*/
  strcpy(makedir, "mkdir /tmp/trac");
  system(makedir);


  /*-------------------------------------------------------*/
  /* Ensure initial program environment is set correctly. */
  /*-------------------------------------------------------*/
  if((fp1 = fopen("crosshair_data", "r"))==NULL)
  {
    printf("File crosshair_data not found!\n");
    exit(1);
  }
  fclose(fp1);

  if((fp1 = fopen("blank_320x499", "r"))==NULL)
  {
    printf("File blank_320x499 not found!\n");
    exit(1);
```

```
}
fclose(fp1);

if((fp2 = fopen("/tmp/trac/start1_flag", "w"))==NULL)
{
  printf("Error opening /tmp/trac/start1_flag!\n");
  exit(1);
}
fputc(off, fp2);
fclose(fp2);

if((fp1 = fopen("/tmp/trac/start2_flag", "w"))==NULL)
{
  printf("Error opening /tmp/trac/start2_flag!\n");
  exit(1);
}
fputc(off, fp1);
fclose(fp1);

if((fp2 = fopen("/tmp/trac/start3_flag", "w"))==NULL)
{
  printf("Error opening /tmp/trac/start3_flag!\n");
  exit(1);
}
fputc(off, fp2);
fclose(fp2);

if((fp1 = fopen("/tmp/trac/EOS_flag", "w"))==NULL)
{
  printf("Error opening /tmp/trac/EOS_flag!\n");
  exit(1);
}
fputc(off, fp1);
fclose(fp1);

if((fp2 = fopen("/tmp/trac/option_flag", "w"))==NULL)
{
  printf("Error opening /tmp/trac/option_flag!\n");
  exit(1);
}
fputc(off, fp2);
fclose(fp2);

if((fp1 = fopen("/tmp/trac/state_flag", "w"))==NULL)
{
  printf("Error opening /tmp/trac/state_flag!\n");
  exit(1);
}
fputc(off, fp1);
fclose(fp1);

if((fp2 = fopen("/tmp/trac/state_info", "w"))==NULL)
{
  printf("Error opening /tmp/trac/state_info!\n");
  exit(1);
}
fputc('r', fp2);
fclose(fp2);

if((fp1 = fopen("/tmp/trac/coords_flag_1", "w"))==NULL)
{
  printf("Error opening /tmp/trac/coords_flag_1!\n");
  exit(1);
}
fputc(off, fp1);
```

```
fclose(fp1);

if((fp2 = fopen("/tmp/trac/coords_flag_2", "w"))==NULL)
{
  printf("Error opening /tmp/trac/coords_flag_2!\n");
  exit(1);
}
fputc(off, fp2);
fclose(fp2);

strcpy(clean_up, "rm -f /tmp/trac/*");

/*-------------------------------------------------*/
/* Prompt user to execute the program 'trac_prgm. */
/*-------------------------------------------------*/
printf("%c\n", '\007');
puts("/*-------------------------------------------------*/");
puts("/* This adaptive correlation tracking algorithm   */");
puts("/* requires the concurrent execution of a second  */");
puts("/* program. To ensure proper synchronization and  */");
puts("/* achieve optimal system performance, this add-   */");
puts("/* itional program must be executed in this sys-   */");
puts("/* tem's foreground mode. Open another command     */");
puts("/* window (Command Tool) and execute the program  */");
puts("/* 'trac_prgm' in directory .../plaw/research.     */");
puts("/*-------------------------------------------------*/");

do
{
  if((fp1 = fopen("/tmp/trac/start1_flag", "r"))==NULL)
  {
    printf("Error opening /tmp/trac/start1_flag!\n");
    exit(1);
  }

  /*-------------------------------------------------*/
  /* Check a file repeatedly till content equals 1. */
  /*-------------------------------------------------*/
  temp = fgetc(fp1);
  while(temp!=on)
  {
    fclose(fp1);
    if((fp1 = fopen("/tmp/trac/start1_flag", "r"))==NULL)
    {
      printf("Error opening /tmp/trac/start1_flag!\n");
      exit(1);
    }
    temp = fgetc(fp1);
  }
  fclose(fp1);

  if((fp2 = fopen("/tmp/trac/start1_flag", "w"))==NULL)
  {
    printf("Error opening /tmp/trac/start1_flag!\n");
    exit(1);
  }
  fputc(off, fp2);
  fclose(fp2);

  /*-------------------------------------------------*/
  /* Display targeting prompt and read keyboard input. */
  /*-------------------------------------------------*/
  printf("%c\n", '\007');
  printf("/*--------------------------------------");
  printf("---------*/\n");
```

```
printf("/* This program is currently operating in");
printf(" the     */\n");
printf("/* free-run mode (only the image field-of");
printf("-views   */\n");
printf("/* are being showned with no target being");
printf(" track-  */\n");
printf("/* ed). The available options are: 't' to");
printf(" activ-  */\n");
printf("/* ate the targeting mode, 's' to skip (b");
printf("y-pass)  */\n");
printf("/* a portion of the image sequence, or 'e");
printf("' to     */\n");
printf("/* exit from this program entirely.      ");
printf("          */\n");
printf("/*-------------------------------------");
printf("---------*/\n");
printf("\nEnter desired option:  ");
gets(string);
state = string[0];
state = tolower(state);
while(state!='e' && state!='t' && state!='s')
{
  printf("%c\n", '\007');
  puts("Target or Exit?");
  printf("Enter first letter: ");
  gets(string);
  state = string[0];
  state = tolower(state);
}

if(state=='s')
{
  printf("%c\n", '\007');
  printf("Number of images to be by-passed:  ");
  gets(string);
  step = atoi(string);
}

/*-----------------------------------------------------*/
/* Pass selected option to 'trac_prgm' and reset flag. */
/*-----------------------------------------------------*/
if((fp1 = fopen("/tmp/trac/state_info", "w"))==NULL)
{
  printf("Error opening /tmp/trac/state_info!\n");
  exit(1);
}
fprintf(fp1, "%c\n", state);
if(state=='s')
{
  fprintf(fp1, "%d\n", step);
}
fclose(fp1);

if((fp2 = fopen("/tmp/trac/state_flag", "w"))==NULL)
{
  printf("Error opening /tmp/trac/state_flag!\n");
  exit(1);
}
fputc(on, fp2);
fclose(fp2);

/*-------------------------------------*/
/* Exit program if last image has been */
/* processed or per user's request.    */
/*-------------------------------------*/
```

```
if((fp1 = fopen("/tmp/trac/EOS_flag", "r"))==NULL)
{
  printf("Error opening /tmp/trac/EOS_flag!\n");
  exit(1);
}
EOS = fgetc(fp1);
if(ferror(fp1))
{
  printf("Error reading from /tmp/trac/EOS_flag!\n");
  exit(1);
}
if(state=='s')
{
  while(EOS!=on)
  {
    fclose(fp1);
    if((fp1 = fopen("/tmp/trac/EOS_flag", "r"))==NULL)
    {
      printf("Error opening /tmp/trac/EOS_flag!\n");
      exit(1);
    }
    EOS = fgetc(fp1);
  }
}
fclose(fp1);

if(EOS==on)
{
  system(clean_up);
  printf("%c\n", '\007');
  printf("/*---------------------------------------");
  printf("---------*/\n");
  printf("/* This program has been terminated eithe");
  printf("r in re- */\n");
  printf("/* sponse to the user's request or becaus");
  printf("e the    */\n");
  printf("/* digitized images for this sequence has");
  printf(" been    */\n");
  printf("/* exhausted.                             ");
  printf("         */\n");
  printf("/*---------------------------------------");
  printf("---------*/\n");
  exit(0);
}

if(state=='t')
{
  if((fp1 = fopen("/tmp/trac/state_flag", "r"))==NULL)
  {
    printf("Error opening /tmp/trac/state_flag!\n");
    exit(1);
  }

  /*-----------------------------------------------*/
  /* Check a file repeatedly till content equals 0. */
  /*-----------------------------------------------*/
  temp = fgetc(fp1);
  while(temp!=off)
  {
    fclose(fp1);
    if((fp1 = fopen("/tmp/trac/state_flag", "r"))==NULL)
    {
      printf("Error opening /tmp/trac/state_flag!\n");
      exit(1);
    }
```

```
    temp = fgetc(fp1);
  }
  fclose(fp1);

  printf("%c\n", '\007');
  printf("/*-------------------------------------");
  printf("-----------*/\n");
  printf("/* To select a target, align the cross-");
  printf("hair of    */\n");
  printf("/* the mouse on the target of interest.");
  printf(" Observe   */\n");
  printf("/* the corresponding x- and y- pixel co");
  printf("ordinates  */\n");
  printf("/* shown on the lower portion of the di");
  printf("splay win- */\n");
  printf("/* dow.  Enter these coordinates at the");
  printf(" prompt    */\n");
  printf("/* below.                                ");
  printf("           */\n");
  printf("/*-------------------------------------");
  printf("-----------*/\n");

  /*-------------------------------------------*/
  /* Retrieve user inputted target coordinates. */
  /*-------------------------------------------*/
  do
  {
    printf("\nEnter the x- pixel coordinate:  ");
    if(gets(temp1)==NULL)
    {
      printf("Error reading in x-coordinate!\n");
      exit(1);
    }
    printf("\nEnter the y- pixel coordinate:  ");
    if(gets(temp2)==NULL)
    {
      printf("Error reading in y-coordinate!\n");
      exit(1);
    }
    x_value = atoi(temp1);
    y_value = atoi(temp2);

    /*-------------------------------------------*/
    /* Comfirmation of inputted target coordinates. */
    /*-------------------------------------------*/
    printf("%c\n", '\007');
    printf("\nTarget coordinates are:\n");
    printf("x = %d, ", x_value);
    printf("y = %d  (Y/N)..... ", y_value);
    gets(string);
    response = string[0];
    response = tolower(response);
    while(response!='y' && response!='n')
    {
      printf("%c\n", '\007');
      printf("\nTarget coordinates are:\n");
      printf("x = %d, ", x_value);
      printf("y = %d  (Y/N)..... ", y_value);
      gets(string);
      response = string[0];
      response = tolower(response);
    }
  }while(response!='y');

  printf("%c\n", '\007');
```

```
printf("/*-------------------------------------");
printf("-----------*/\n");
printf("/* The following image processing techn");
printf("iques can  */\n");
printf("/* be applied to this adaptive correlat");
printf("ion track- */\n");
printf("/* ing algorithm .                      ");
printf("           */\n");
printf("/*                                      ");
printf("           */\n");
printf("/*   A...Binarization normalized to 255");
printf(".          */\n");
printf("/*                                      ");
printf("           */\n");
printf("/*   B...Binarized with image gray-scal");
printf("e added.   */\n");
printf("/*                                      ");
printf("           */\n");
printf("/*   C...Truthed target regions. (From ");
printf("8-1Km)     */\n");
printf("/*                                      ");
printf("           */\n");
printf("/*   D...Truthed template w/ unmodified");
printf(" scene.    */\n");
printf("/*       (Valid from 8-1Km)             ");
printf("           */\n");
printf("/*                                      ");
printf("           */\n");
printf("/*   E...Blended images. (Binarized)    ");
printf("           */\n");
printf("/*                                      ");
printf("           */\n");
printf("/*   F...Blended images. (Gray-scaled a");
printf("dded)      */\n");
printf("/*                                      ");
printf("           */\n");
printf("/*   G...Image correlation without enha");
printf("ncements.  */\n");
printf("/*                                      ");
printf("           */\n");
printf("/*   H...Optical adaptive tracking simu");
printf("lation.    */\n");
printf("/*                                      ");
printf("           */\n");
printf("/*-------------------------------------");
printf("-----------*/\n");
printf("\nEnter desired option:  ");
gets(string);
it = string[0];
it = tolower(it);
while(it!='a' && it!='b' && it!='c' && it!='d' && it!='e' && it!='f' && it!='g' && it!='h')
{
  printf("%c\n", '\007');
  printf("Enter option A, B, C, D, E, F, G, or H... ");
  gets(string);
  it = string[0];
  it = tolower(it);
}

if(it=='a' || it=='b' || it=='e' || it=='f')
{
  printf("%c\n", '\007');
  printf("Binarize via:  A) Line-by-line average\n");
  printf("               B) Scene average\n");
  printf("               or C) Cline Binarization?... ");
```

```
  gets(string);
  binary = string[0];
  binary = tolower(binary);
  while(binary!='a' &&  binary!='b' && binary!='c')
  {
    printf("%c\n", '\007');
    printf("Enter option A, B, or C...  ");
    gets(string);
    binary = string[0];
    binary = tolower(binary);
  }
  if(binary=='a')  binary = 'z';

  printf("%c\n", '\007');
  printf("/*-------------------------------------");
  printf("------------*/\n");
  printf("/* The binarization threshold for this ");
  printf("program is */\n");
  printf("/* computed as 'threshold+(factor x thr");
  printf("eshold)'.  */\n");
  printf("/* To vary the threshold, enter a value");
  printf(" for the   */\n");
  printf("/* variable 'factor' (value must be bet");
  printf("ween -1    */\n");
  printf("/* and 1).                              ");
  printf("          */\n");
  printf("/*-------------------------------------");
  printf("------------*/\n");
  printf("Enter a value between 0 and 1: ...  ");
  gets(string);
  factor = atof(string);
}

printf("%c\n", '\007');
printf("Use edge extractor (y or n)? \n");
gets(string);
edger = string[0];
edger = tolower(edger);


printf("%c\n", '\007');
printf("/*-------------------------------------");
printf("-----------*/\n");
printf("/* The template and scene sizes can als");
printf("o be chan- */\n");
printf("/* ged. The available sizes are:       ");
printf("          */\n");
printf("/*                                     ");
printf("          */\n");
printf("/*    L...Template(64x40) and Scene(192");
printf("x120)     */\n");
printf("/*                                     ");
printf("          */\n");
printf("/*    M...Template(42x28) and Scene(116");
printf("x72)      */\n");
printf("/*                                     ");
printf("          */\n");
printf("/*    S...Template(30x19) and Scene(60x");
printf("38)       */\n");
printf("/*                                     ");
printf("          */\n");
printf("/*    A...Adaptive Template and Small S");
printf("cene      */\n");
printf("/*                                     ");
printf("          */\n");
```

```
printf("/*    F...Adaptive Template and Full Sc");
printf("ene        */\n");
printf("/*                                      ");
printf("           */\n");
printf("/*--------------------------------------");
printf("-----------*/\n");
printf("\nEnter desired option:  ");
gets(string);
size = string[0];
size = tolower(size);
while(size!='l' && size!='m' && size!='s' && size!='a' && size!='f')
{
  printf("%c\n", '\007');
  printf("Enter option L, M, S, A, or F.....  ");
  gets(string);
  size = string[0];
  size = tolower(size);
}

if((fp2 = fopen("/tmp/trac/coords_info_1", "w"))==NULL)
{
  printf("Error opening /tmp/trac/coords_info_1!\n");
  exit(1);
}
fprintf(fp2, "%d\n%d\n%c\n", x_value, y_value, it);
fprintf(fp2, "%f\n%c\n", factor, size);
fprintf(fp2, "%c\n%c\n", binary, edger);
fclose(fp2);

if((fp1 = fopen("/tmp/trac/coords_flag_1", "w"))==NULL)
{
  printf("Error opening /tmp/trac/coords_flag_1!\n");
  exit(1);
}
fputc(on, fp1);
fclose(fp1);

if((fp2 = fopen("/tmp/trac/coords_flag_2", "r"))==NULL)
{
  printf("Error opening /tmp/trac/coords_flag_2!\n");
  exit(1);
}

/*-------------------------------------------------*/
/* Check a file repeatedly till content equals 1. */
/*-------------------------------------------------*/
temp = fgetc(fp2);
while(temp!=on)
{
  fclose(fp2);
  if((fp1 = fopen("/tmp/trac/coords_flag_2", "r"))==NULL)
  {
    printf("Error opening /tmp/trac/coords_flag_2!\n");
    exit(1);
  }
  temp = fgetc(fp2);
}
fclose(fp2);

if((fp1 = fopen("/tmp/trac/coords_flag_2", "w"))==NULL)
{
  printf("Error opening /tmp/trac/coords_flag_2!\n");
  exit(1);
}
fputc(off, fp1);
```

```
fclose(fp1);

/*-------------------------------------------------*/
/* Retrieve range, target number, and true center-of- */
/* target coordinates from the program 'trac_prgm'.   */
/*-------------------------------------------------*/
if((fp2 = fopen("/tmp/trac/coords_info_2", "r"))==NULL)
{
  puts("Error reading from /tmp/trac/coords_info_2!");
  exit(1);
}
temp = fgetc(fp2);
if(temp!=eff)
{
  fgets(string, 128, fp2);
  fgets(string, 128, fp2);
  x_coord = atof(string);
  fgets(string, 128, fp2);
  y_coord = atof(string);
  fgets(string, 128, fp2);
  character = atoi(string);
  fclose(fp2);

  /*-----------------------------------*/
  /* Give the user the option to track */
  /* on the true center-of-target.     */
  /*-----------------------------------*/
  printf("%c\n", '\007');
  printf("********** TARGET RANGE: %d", character);
  printf(" meters **********\n");
  printf("Inputted target coordinates");
  printf(" were:  x = %d, y = %d\n", x_value, y_value);
  printf("True center-of-target is at:  ");
  printf("x = %5.1f, y = %5.1f\n\n", x_coord, y_coord);
  printf("Track on center-of-target instead?");
  printf("   (Y/N)..... ");
  gets(string);
  response = string[0];
  response = tolower(response);
  while(response!='y' && response!='n')
  {
    printf("%c\n", '\007');
    printf("\nTarget coordinates are:\n");
    printf("x = %d, ", x_value);
    printf("y = %d  (Y/N)..... ", y_value);
    gets(string);
    response = string[0];
    response = tolower(response);
  }

  /*-----------------------------------------*/
  /* Pass user's response to program 'trac_prgm'. */
  /*-----------------------------------------*/
  if((fp2 = fopen("/tmp/trac/coords_info_1", "w"))==NULL)
  {
    printf("Error opening /tmp/trac/coords_info_1!\n");
    exit(1);
  }
  fprintf(fp2, "%c\n", response);
  fclose(fp2);

  if((fp1 = fopen("/tmp/trac/coords_flag_1", "w"))==NULL)
  {
    printf("Error opening /tmp/trac/coords_flag_1!\n");
    exit(1);
```

```
  }
  fputc(on, fp1);
  fclose(fp1);
}

do
{
  if((fp2 = fopen("/tmp/trac/start2_flag", "r"))==NULL)
  {
    printf("Error opening /tmp/trac/start2_flag!\n");
    exit(1);
  }

  /*-----------------------------------------------*/
  /* Check a file repeatedly till content equals 1. */
  /*-----------------------------------------------*/
  temp = fgetc(fp2);
  while(temp!=on)
  {
    fclose(fp2);
    if((fp2 = fopen("/tmp/trac/start2_flag", "r"))==NULL)
    {
      printf("Error opening /tmp/trac/start2_flag!\n");
      exit(1);
    }
    temp = fgetc(fp2);
  }
  fclose(fp2);

  if((fp1 = fopen("/tmp/trac/start2_flag", "w"))==NULL)
  {
    printf("Error opening /tmp/trac/start2_flag!\n");
    exit(1);
  }
  fputc(off, fp1);
  fclose(fp1);

  printf("%c\n", '\007');
  printf("/*-----------------------------------");
  printf("-------------*/\n");
  printf("/* This program is currently tracking");
  printf(" the chosen  */\n");
  printf("/* target-of-interest using adaptive ");
  printf("correlation. */\n");
  printf("/* Available options are:            ");
  printf("             */\n");
  printf("/*                                   ");
  printf("             */\n");
  printf("/*  W...Vary the template and scene w");
  printf("indow sizes. */\n");
  printf("/*                                   ");
  printf("             */\n");
  printf("/*  B...Change the binarization techn");
  printf("ique.        */\n");
  printf("/*                                   ");
  printf("             */\n");
  printf("/*  T...Vary the binarization thresho");
  printf("ld.          */\n");
  printf("/*                                   ");
  printf("             */\n");
  printf("/*  N...Select a different target-of-");
  printf("interest.    */\n");
  printf("/*                                   ");
  printf("             */\n");
  printf("/*  E...Exit from this program entire");
```

```
printf("ly.          */\n");
printf("/*                                        ");
printf("             */\n");
printf("/*-----------------------------------");
printf("-------------*/\n");
printf("\nEnter desired option: ");
gets(string);
opt = string[0];
opt = tolower(opt);

/*-------------------------------------------------*/
/* Check repeatedly till a valid option was chosen. */
/*-------------------------------------------------*/
while(opt!='e' && opt!='w' && opt!='n' && opt!='t' && opt!='b')
{
  printf("%c\n", '\007');
  puts("Window, Binary, Threshold, New, or Exit?");
  printf("Enter the first letter of choice: ");
  gets(string);
  opt = string[0];
  opt = tolower(opt);
}
choice = opt;

if(opt=='w')
{
  printf("%c\n", '\007');
  puts("Small, Medium, Large, Adapt/Small, or Full/Adapt.");
  printf("Enter the first letter of your choice:  ");
  gets(string);
  choice = string[0];
  choice = tolower(choice);
  while(choice!='s' &&  choice!='m' && choice!='l' && choice!='a' && choice!='f')
  {
    printf("%c\n", '\007');
    printf("Enter option S, M, L, A, or F...  ");
    gets(string);
    choice = string[0];
    choice = tolower(choice);
  }
  opt = 'o';
}

if(opt=='t')
{
  printf("%c\n", '\007');
  printf("Please enter a value between 0 and 1... ");
  gets(string);
  factor = atof(string);
  opt = 'o';
}

if(opt=='b')
{
  printf("%c\n", '\007');
  printf("A) Line-by-line, B) Scene, or C) Cline?...  ");
  gets(string);
  choice = string[0];
  choice = tolower(choice);
  while(choice!='a' &&  choice!='b' && choice!='c')
  {
    printf("%c\n", '\007');
    printf("Enter option A, B, or C...  ");
    gets(string);
    choice = string[0];
```


85

```
    choice = tolower(choice);
   }
   if(choice=='a')  choice = 'z';
   opt = 'o';
 }

 if((fp2 = fopen("/tmp/trac/option_info", "w"))==NULL)
 {
   printf("Error opening /tmp/trac/option_info!\n");
   exit(1);
 }
 fprintf(fp2, "%c\n", choice);
 if(choice=='t')
 {
   fprintf(fp2, "%f\n", factor);
 }
 fclose(fp2);

 if((fp1 = fopen("/tmp/trac/option_flag", "w"))==NULL)
 {
   printf("Error opening /tmp/trac/option_flag!\n");
   exit(1);
 }
 fputc(on, fp1);
 fclose(fp1);

 /*-------------------------------------*/
 /* Exit program if last image has been */
 /* processed or per user's request.    */
 /*-------------------------------------*/
 if((fp1 = fopen("/tmp/trac/EOS_flag", "r"))==NULL)
 {
   printf("Error opening /tmp/trac/EOS_flag!\n");
   exit(1);
 }
 EOS = fgetc(fp1);
 if(ferror(fp1))
 {
   printf("Error reading from /tmp/trac/EOS_flag!\n");
   exit(1);
 }
 if(opt=='o')
 {
   while(EOS!=on)
   {
     fclose(fp1);
     if((fp1 = fopen("/tmp/trac/EOS_flag", "r"))==NULL)
     {
       printf("Error opening /tmp/trac/EOS_flag!\n");
       exit(1);
     }
     EOS = fgetc(fp1);
   }
 }
 fclose(fp1);

 if(EOS==on)
 {
   system(clean_up);
   printf("%c\n", '\007');
   printf("/*-------------------------------------");
   printf("----------*/\n");
   printf("/* This program has been terminated eithe");
   printf("r in re- */\n");
   printf("/* sponse to the user's request or becaus");
```

```
            printf("e the    e/\n");
            printf("/e digitized images for this sequence has");
            printf(" been    e/\n");
            printf("/e exhausted.                              ");
            printf("         e/\n");
            printf("/e---------------------------------------");
            printf("---------e/\n");
            exit(0);
          }
        }while(opt=='e');
      }
    }while(opt=='n');
}
```

B.2 TRAC_PRGM.C

```
/*****************************************************************/
/* TRAC_PROM.C                                                   */
/*                                                               */
/*            AIR FORCE INSTITUTE OF TECHNOLOGY                  */
/* TRACKS TARGET USING THE TECHNIQUE OF ADAPTIVE CORRELATION     */
/*                     by Capt Paul D. Law                       */
/*                       October 7, 1991                         */
/*                                                               */
/*                         modified by                           */
/*                   Capt Dennis A. Montera                      */
/*                     September 1, 1992                         */
/*                                                               */
/* This is the second of a pair of programs which must be ran    */
/* in unison in order to implement the adaptive correlation      */
/* tracking algorithm. It performs the required computations,    */
/* manages files, and utilizes KHOROS image processing rout-     */
/* ines to manipulate the sequential FLIR images. In general,    */
/* it is responsible for the overall control of the adaptive     */
/* correlation tracking process.                                 */
/*****************************************************************/

#include <stdio.h>
#include <stdlib.h>
#include <sys/stat.h>
#include <math.h>
#include <time.h>

void line_bin();

void main(argc, argv)
int argc;
char *argv[];
{
  /*-----------------------------------------*/
  /* Define and initialize program variables. */
  /*-----------------------------------------*/
  FILE *fp1, *fp2, *fp3, *fp4, *fp5, *fp6, *fp7, *fp8;
  char r2v_cs[128], pad3_2x5[128], delay[128], byte_cs[128];
  char display_fov[128], pad4x8_cs[128], rename0[128];
  char pad4x8_ps[128], extr_s1[256], extr_t1[256], fft_s1[128];
  char fft_t1[128], display_s1[128], r2v_ps[128], byte_s1[128];
  char pad256_s1[128], pad256_t1[128], display_t1[128];
  char conjug_t1[128], inv_fft[128], byte_flip[128], term1[15];
  char extr_sub[256], byte_sub[128], byte_fov[128], state_flag;
  char mode = 'f', state = 'r', opt = 'a', byte_t1[128];
  char add_cross[128], files[200][128], b_extru2[128], set;
  char ps_value[12], k_displ_fov[128], k_displ_t1[128], temp;
  char q_flip128[128], string[128], size, k_displ_s1[128];
  char search[128], mthd, intru_t1[128], r2v_old[128], corr_min[128];
  char mean_cs[128], mean_ps[128], copy_t1[128], copy_s1[128];
  char stat_pk[250], opt_flag, multi[128], on = '1', off = '0';
  char filename[128], tru_cntr[200][128], run_num[15], Cstat[250];
  char cs_stat[250], d_s1[128], t1_4stat[128], s1_4stat[128];
  char extru_s1[128], extru_t1[128], intru_s1[128], a_ins[128];
  char ps_stat[250], b_ins[128], gt1_4stat[128], gs1_4stat[128];
  char pad128_s1[128], pad128_t1[128], q_flip64[128];
  char bin_s1[128], bin_t1[128], s1_x_bin[128], t1_x_bin[128];
  char pad_4displ[128], blend_t1[128], blend_s1[128];
  char hpf_s1[128], hpf_t1[128], pad64_t1[128], pad64_s1[128];
  char stat_t1blend[250], stat_s1blend[250], blend_x_t1[128];
```

```
char blend_x_s1[128], cmplx_t1[128], cmplx_s1[128], term2[15];
char ct1_4stat[128], cs1_4stat[128], a_pad4x8[128], rename4[128];
char corr_var[128], corr_stdev[128], corr_mean[128], corr_max[128];
char q_flip32[128], t1_2real[128], s1_2real[128], a_pad3[128];
char b_pad4x8[128], c_pad4x8[128], a_extru2[128], str0[128];
char clr_scr[128], summary[128], str1[128], str2[128];
char d_pad4x8[128], a_2asc1[128], a_2asc2[128], a_2asc3[128];
char b_2asc1[128], b_2asc2[128], b_2asc3[128], a_2byte1[128];
char b_2byte1[128], a_extru1[128], b_extru1[128], a_pad1[128];
char b_pad1[128], a_extr1[128], b_extr1[128], a_2viff[128];
char b_2viff[128], a_extr2[128], c_pad1_4x8[128], b_pad3[128];
char d_pad1_4x8[128], b_extr2[128], targ_file[128], rename3[128];
char Cmean[15], Cvar[15], Cmax[15], Cmin[15], Cstdev[15], c2real[128];
char a_multi[128], b_multi[128], copy_cs[128], copy_ps[128];
char rbin_t1[128], rbin_s1[128], a_2byte2[128], b_2byte2[128];
char bin_state = 'b', cs_pad4x8[128], ps_pad4x8[128], end[5];
char s_4mean[128], t_4mean[128], stat_cs[250], stat_ps[250];
char hi_ps[128], hi_cs[128], name1[128], dyn1_s1[128];
char dyn1_t1[128], and_s1[128], and_t1[128], dyn2_s1[128];
char dyn2_t1[128], cpy_cs[128], name2[128], stat_st1[250];
char rename2[128], plotname[128], stat_ss1[250], rename1[128];
char displ_file[128], saveimg[128], term3[50], padt1[128];
char corrtt[300], corrts[300], display = 'g';
char extr_cross[128], pad_cross[128], cut_cross[128], tempo[128];
char pad512_s1[128], pad512_t1[128], fft_t2[128];
char extr_win[128], edger = 'n';
int cycle = 0, pass = 0, count = 0, run = 0, character, r;
int x_value, y_value, x_term, y_term, targ_num, height, width;
int integ1, integ2, index = 0, curr_x, curr_y, curr_h, s;
int curr_w, old_x_temp1, old_y_temp1, old_x_scene1, step;
int old_y_scene1, x_numb, y_numb, max, min, tt_max, tt_min;
int st_max, st_min, tb_max, tb_min, total, sum, value;
int swap, x_coord, y_coord, i, j, k, l, m, n, x_peak, y_peak;
int pad_size = 128, offset = 25, length, fr_info, tr_info;
int x_scene1, y_scene1, x_template1, y_template1, limit;
int y_template1_size, x_template1_size, y_scene1_size, digit3;
int x_scene1_size, y_display_size = 200, x_display_size = 310;
int x_display, y_display, times = 0, job1, job2, job3 = 0;
int high_o_ps, high_o_cs, number, cond1, cond2, digit1, digit2;
int xy[499][320], pointer, xcen, ycen, cxx, cyy, txx, tyy, fact;
int a1, a2, b1, b2, c1, c2, d1, d2, xx, yy;
int tmpx, tmpy, cownt, xpnt, ypnt, xlen, ylen, xedge, yedge;
int xtar[20], ytar[20], holdx[20], holdy[20], permx, permy;
float blend_mean, s1_mean, t1_mean, factor = 0.15;
float cortemp[7][7], cordata[499][320], tempdata;
float holder2, norm1, norm2, norm3, normt, normc;
double result, distance, mean, numb1, numb2, angle;
double pie = 3.141592653589793;
double t_value, power, npower, tt_npower, tt_power;
double st_npower, st_power, tb_power, tb_npower, mean_o_ps;
double mean_o_cs, t_npr, t_pr, s_npr, s_pr, pr, npr, error;
double xf_term, yf_term, xf_coord, yf_coord, f_temp, xf_peak;
double yf_peak, fvalue1, fvalue2, xf_numb, yf_numb;
double xf_cntr, yf_cntr, out1, out2, out3, out4, out5, out6;
double out7, out8, temp_np, temp_p, scen_np, scen_p, factor2;
double corrvar, corrmean, corrstdev, corrmax, corrmin, check;
double percent, holder, dist, magerr[20];
double normal1, normal2, normal3, normalc, normalt;

/*-------------------------------------------------*/
/* Checks for base filename of the image sequence */
/* and ensure 'trac_menu' has been executed.      */
/*-------------------------------------------------*/
if((fp1 = fopen("/tmp/trac/start1_flag", "r"))==NULL)
{
```

```
  printf("%c\n", '\007');
  puts("/*-------------------------------------------*/");
  puts("/* This is the second of two programs which   */");
  puts("/* must be executed in unison in order to im- */");
  puts("/* plement the adaptive correlation tracking  */");
  puts("/* algorithm. To ensure correct synchronizat-  /");
  puts("/* ion, the first program 'trac_menu' must be */");
  puts("/* running in the system's foreground mode    */");
  puts("/* prior to the execution of this program.    */");
  puts("/* Open another command window (Command Tool) */");
  puts("/* and execute 'trac_menu' located in the     */");
  puts("/* .../plaw/research directory before attemp- */");
  puts("/* ting to re-execute this program.           */");
  puts("/*-------------------------------------------*/");
  printf("\n");
  exit(1);
}
if(argc<2)
{
  printf("%c\n", '\007');
  puts("/*-------------------------------------------*/");
  puts("/* Incorrect command syntax or missing base  */");
  puts("/* filename! The correct form of the command  */");
  puts("/* is COMMAND BASENAME*.EXT where basename is */");
  puts("/* the portion of the filename which is com-  */");
  puts("/* mon to a group of image files. For example */");
  puts("/* the base filename for a sequence of image  */");
  puts("/* files with names 0b3010.pix to 0b3080.pix  */");
  puts("/* is 0b3*.pix. Re-execute this program with  */");
  puts("/* the correct syntax and base filename.      */");
  puts("/*-------------------------------------------*/");
  printf("\n");
  exit(1);
}
fclose(fp1);

/*-----------------------------------*/
/* Determine if displays to screen - */
/* must be 'no' to run in background */
/*-----------------------------------*/
while(display!='y' && display !='n')
{
  printf("%c\n", '\007');
  puts("Displays on screen (Y or N)?");
  gets(string);
  display = string[0];
  display = tolower(display);
}

for (xx = 0; xx <= 19; ++ xx) {
  holdx[xx] = 0;
  holdy[xx] = 0;
  xtar[xx] = 0;
  ytar[xx] = 0;
}


/*-----------------------------------*/
/* Determine name of file containing */
/* true target location information. */
/*-----------------------------------*/
strcpy(string, argv[1]);
length = strlen(string);
for(i=0; i<(length - 10); i++)
{
```

```
  filename[i] = string[i];
}
for(k=0; k<(length - 13); k++)
{
  name1[k] = string[k];
}
name1[k] = '\0';
for(j=0; j<3; j++)
{
  name2[j] = string[k + j];
}
name2[j] = '\0';

for(j=0; j<3; j++)  run_num[j] = filename[i - 3 + j];
run_num[j] = '\0';
strcat(filename, "_tru_cntr");

if((fp2 = fopen(filename, "r"))==NULL)
{
  printf("Error reading from %s!\n", filename);
  exit(1);
}
fclose(fp2);

/*-------------------------------------------*/
/* Determine target range from image filename. */
/*-------------------------------------------*/

for(i=0; i<(argc - 1); i++)
{
  strcpy(string, argv[argc - (i + 1)]);
  length = strlen(string);
  for(j=0; j<5; j++)  term1[j] = string[length - 9 + j];
  term1[j] = '\0';
  fr_info = atoi(term1);

  /*---------------------------------------*/
  /* Search for matching range information */
  /* in the image truth data file.         */
  /*---------------------------------------*/
  sprintf(search, "grep ' %d ' %s", fr_info, filename);
  strcat(search, " > /tmp/trac/temp_txt");
  system(search);

  if((fp4 = fopen("/tmp/trac/temp_txt", "r"))==NULL)
  {
    printf("Error reading /tmp/trac/temp_txt!\n");
    exit(1);
  }

  /*---------------------------------------*/
  /* If found, store the target locations,  */
  /* else store the predetermined number 0. */
  /*---------------------------------------*/
  if(fgets(search, 128, fp4)==NULL)
  {
    strcpy(search, term1);
    strcat(search, "     0.0 ,  0.0  ;  0 , 0  ;  0 ;  0");
    strcat(search, "     0.0 ,  0.0  ;  0 , 0  ;  0 ;  0");
    strcat(search, "     0.0 ,  0.0  ;  0 , 0  ;  0 ;  0\n");
  }

  for(j=0; j<128; j++)
  {
    files[i][j] = string[j];
```

```
    tru_cntr[i][j] = search[j];
  }
  fclose(fp4);
}

/*-------------------------------------------------*/
/* Initialize KHOROS sub-routine command strings. */
/*-------------------------------------------------*/
strcpy(r2v_cs, "raw2viff -i /tmp/trac/current_scene ");
strcat(r2v_cs, "-o /tmp/trac/scene1_viff -l 0 -t byte -mt 'Local Machine' ");
strcat(r2v_cs, "-r 320 -c 499");

strcpy(pad3_2x5, "vpad -i /tmp/trac/scene1_viff ");
strcat(pad3_2x5, "-o /tmp/trac/scene1_pad_3.2x5 -r 320 ");
strcat(pad3_2x5, "-c 500 -d 0 -e 0 -j 0 -k 0");

strcpy(delay, "vresize -i /tmp/trac/scene1_pad_3.2x5 ");
strcat(delay, "-o /tmp/trac/scene1_resize -h 1 -v 1");

strcpy(byte_fov, "vconvert -i /tmp/trac/scene1_resize ");
strcat(byte_fov, "-o /tmp/trac/FOV_displ -n 255 ");
strcat(byte_fov, "-t byte -b 0");

strcpy(display_fov, "putimage -i /tmp/trac/FOV_displ ");
strcat(display_fov, "-x 408 -y 0 -update .01&");

strcpy(r2v_ps, "raw2viff -i /tmp/trac/previous_scene ");
strcat(r2v_ps, "-o /tmp/trac/temp1_viff -l 0 -t byte -mt 'Local Machine' ");
strcat(r2v_ps, "-r 320 -c 499");

strcpy(r2v_old, "raw2viff -i /tmp/trac/old_scene ");
strcat(r2v_old, "-o /tmp/trac/old_viff -l 0 -t byte -mt 'Local Machine' ");
strcat(r2v_old, "-r 320 -c 499");

strcpy(pad4x8_cs, "vpad -i /tmp/trac/scene1_viff ");
strcat(pad4x8_cs, "-o /tmp/trac/scene1_pad_4x8 -r 400 ");
strcat(pad4x8_cs, "-c 800 -d 40 -e 150 -j 0 -k 0");

strcpy(pad4x8_ps, "vpad -i /tmp/trac/temp1_viff ");
strcat(pad4x8_ps, "-o /tmp/trac/temp1_pad_4x8 -r 400 -c 800 ");
strcat(pad4x8_ps, "-d 40 -e 150 -j 0 -k 0");

strcpy(cs_pad4x8, "vpad -i /tmp/trac/scene1_4stat ");
strcat(cs_pad4x8, "-o /tmp/trac/pad_4x8_scene1 -r 400 ");
strcat(cs_pad4x8, "-c 800 -d 40 -e 150 -j 0 -k 0");

strcpy(ps_pad4x8, "vpad -i /tmp/trac/temp1_4stat ");
strcat(ps_pad4x8, "-o /tmp/trac/pad_4x8_temp1 -r 400 -c 800 ");
strcat(ps_pad4x8, "-d 40 -e 150 -j 0 -k 0");

strcpy(a_pad4x8, "vpad -i /tmp/trac/a_pad1_w0s ");
strcat(a_pad4x8, "-o /tmp/trac/a_pad2_4x8 -r 400 ");
strcat(a_pad4x8, "-c 800 -d 40 -e 150 -j 0 -k 0");

strcpy(b_pad4x8, "vpad -i /tmp/trac/b_pad1_w0s ");
strcat(b_pad4x8, "-o /tmp/trac/b_pad2_4x8 -r 400 ");
strcat(b_pad4x8, "-c 800 -d 40 -e 150 -j 0 -k 0");

strcpy(c_pad4x8, "vpad -i /tmp/trac/a_byte_intemp ");
strcat(c_pad4x8, "-o /tmp/trac/c_pad1_4x8 -r 400 ");
strcat(c_pad4x8, "-c 800 -d 40 -e 150 -j 0 -k 0");

strcpy(d_pad4x8, "vpad -i /tmp/trac/b_byte_inscene ");
strcat(d_pad4x8, "-o /tmp/trac/d_pad1_4x8 -r 400 ");
strcat(d_pad4x8, "-c 800 -d 40 -e 150 -j 0 -k 0");
```

```
strcpy(pad_4displ, "vpad -i /tmp/trac/scene1_viff ");
strcat(pad_4displ, "-o /tmp/trac/pad_4x8_displ -r 400 -c 800 ");
strcat(pad_4displ, "-d 40 -e 150 -j 0 -k 0");

strcpy(display_t1, "putimage -i /tmp/trac/T1_disp ");
strcat(display_t1, "-x 931 -y 60 -update .01&");

strcpy(display_s1, "putimage -i /tmp/trac/sub_scene1 ");
strcat(display_s1, "-x 931 -y 200 -update .01&");

strcpy(pad256_t1, "vpad -i /tmp/trac/sub_temp1 ");
strcat(pad256_t1, "-o /tmp/trac/sub_temp1_pad -r 256 -c 256 ");
strcat(pad256_t1, "-d 108 -e 96 -j 0 -k 0");

strcpy(pad128_t1, "vpad -i /tmp/trac/sub_temp1 ");
strcat(pad128_t1, "-o /tmp/trac/sub_temp1_pad -r 128 -c 128 ");
strcat(pad128_t1, "-d 50 -e 43 -j 0 -k 0");

strcpy(pad64_t1, "vpad -i /tmp/trac/sub_temp1 ");
strcat(pad64_t1, "-o /tmp/trac/sub_temp1_pad -r 64 -c 64 ");
strcat(pad64_t1, "-d 22 -e 17 -j 0 -k 0");

strcpy(pad256_s1, "vpad -i /tmp/trac/sub_scene1 ");
strcat(pad256_s1, "-o /tmp/trac/sub_scene1_pad -r 256 ");
strcat(pad256_s1, "-c 256 -d 68 -e 32 -j 0 -k 0");

strcpy(pad128_s1, "vpad -i /tmp/trac/sub_scene1 ");
strcat(pad128_s1, "-o /tmp/trac/sub_scene1_pad -r 128 ");
strcat(pad128_s1, "-c 128 -d 28 -e 6 -j 0 -k 0");

strcpy(pad64_s1, "vpad -i /tmp/trac/sub_scene1 ");
strcat(pad64_s1, "-o /tmp/trac/sub_scene1_pad -r 64 ");
strcat(pad64_s1, "-c 64 -d 13 -e 2 -j 0 -k 0");

strcpy(pad512_s1, "vpad -i /tmp/trac/sub_scene1 ");
strcat(pad512_s1, "-o /tmp/trac/sub_scene1_pad -r 512 ");
strcat(pad512_s1, "-c 512 -d 0 -e 0 -j 0 -k 0");

strcpy(fft_t1, "vfft -i1 /tmp/trac/sub_temp1_pad ");
strcat(fft_t1, "-o1 /tmp/trac/sub_temp1_fft -d 0");

strcpy(fft_t2, "vfft -i1 /tmp/trac/sub_temp2_pad ");
strcat(fft_t2, "-o1 /tmp/trac/sub_temp2_fft -d 0");

strcpy(fft_s1, "vfft -i1 /tmp/trac/sub_scene1_pad ");
strcat(fft_s1, "-o1 /tmp/trac/sub_scene1_fft -d 0");

strcpy(conjug_t1, "vconj -i /tmp/trac/sub_temp1_fft ");
strcat(conjug_t1, "-o /tmp/trac/sub_temp1_conj");

strcpy(multi, "vmul -i1 /tmp/trac/sub_scene1_fft ");
strcat(multi, "-i2 /tmp/trac/sub_temp1_conj ");
strcat(multi, "-o /tmp/trac/scene1_x_temp1");

strcpy(corrts, "vmul -i1 /tmp/trac/sub_scene1_fft ");
strcat(corrts, "-i2 /tmp/trac/sub_temp1_conj ");
strcat(corrts, "-o - | ");
strcat(corrts, "vfft -i1 - ");
strcat(corrts, "-o1 - -d 1 | ");
strcat(corrts, "vtranslat -i - ");
strcat(corrts, "-o - -x 256 -y 256 -w 1 | ");
strcat(corrts, "vctor -i - ");
strcat(corrts, "-o - -t 3 | ");
strcat(corrts, "vextract -i - -o - ");
```


93

```
strcat(corrts, "-x 0 -y 0 -w 499 -h 320 | ");
strcat(corrts, "viff2mat -i - ");
strcat(corrts, "-o /tmp/trac/tmp_3 -j 1 -b 0 ");
strcat(corrts, "-d 0 -f 1 ");

strcpy(corrtt, "vmul -i1 /tmp/trac/sub_temp2_fft ");
strcat(corrtt, "-i2 /tmp/trac/sub_temp1_conj ");
strcat(corrtt, "-o - | ");
strcat(corrtt, "vfft -i1 - ");
strcat(corrtt, "-o1 - -d 1 | ");
strcat(corrtt, "vtranslat -i - ");
strcat(corrtt, "-o - -x 256 -y 256 -w 1 | ");
strcat(corrtt, "vctor -i - ");
strcat(corrtt, "-o - -t 3 | ");
strcat(corrtt, "vextract -i - -o - ");
strcat(corrtt, "-x 253 -y 253 -w 7 -h 7 | ");
strcat(corrtt, "viff2mat -i - ");
strcat(corrtt, "-o /tmp/trac/tmp_3 -j 1 -b 0 ");
strcat(corrtt, "-d 0 -f 1 ");

strcpy(b_multi, "vmul -i1 /tmp/trac/sviff_4stat ");
strcat(b_multi, "-i2 /tmp/trac/b_padi_wis ");
strcat(b_multi, "-o /tmp/trac/b_byte_inscene");

strcpy(a_multi, "vmul -i1 /tmp/trac/tviff_4stat ");
strcat(a_multi, "-i2 /tmp/trac/a_padi_wis ");
strcat(a_multi, "-o /tmp/trac/a_byte_intemp");

strcpy(inv_fft, "vfft -i1 /tmp/trac/scenei_x_temp1 ");
strcat(inv_fft, "-o1 /tmp/trac/inverse_fft_1 -d 1");

strcpy(q_flip128, "vtranslat -i /tmp/trac/inverse_fft_1 ");
strcat(q_flip128, "-o /tmp/trac/vtranslate_1 -x 128 -y 128 -w 1");

strcpy(q_flip64, "vtranslat -i /tmp/trac/inverse_fft_1 ");
strcat(q_flip64, "-o /tmp/trac/vtranslate_1 -x 64 -y 64 -w 1");

strcpy(q_flip32, "vtranslat -i /tmp/trac/inverse_fft_1 ");
strcat(q_flip32, "-o /tmp/trac/vtranslate_1 -x 32 -y 32 -w 1");

strcpy(byte_flip, "vconvert -i /tmp/trac/vtranslate_1 ");
strcat(byte_flip, "-o /tmp/trac/conv2byte_1 -n 255 ");
strcat(byte_flip, "-t byte -b 0");

strcpy(stat_pk, "vstats -i /tmp/trac/conv2byte_1 ");
strcat(stat_pk, "-f /tmp/trac/peak_coords_1 -all 0 -mu 0 ");
strcat(stat_pk, "-var 0 -sd 0 -rms 0 -vmax 0 -xmax 1 ");
strcat(stat_pk, "-ymax 1 -vmin 0 -xmin 0 -ymin 0 -in 0 ");
strcat(stat_pk, "-pin 0 -nin 0 -pts 0 -ppts 0 -npts 0 ");
strcat(stat_pk, "-sk 0 -kur 0 -ent 0 -con 0 -b 1");

strcpy(byte_sub, "vconvert -i /tmp/trac/sub_displ_conv ");
strcat(byte_sub, "-o /tmp/trac/ready_4displ -n 255 ");
strcat(byte_sub, "-t byte -b 0");

strcpy(add_cross, "vadd -i1 /tmp/trac/sub_displ_data ");
strcat(add_cross, "-i2 crosshair_data ");
strcat(add_cross, "-o /tmp/trac/sub_displ_conv");

strcpy(extr_cross, "vextract -i crosshair_data ");
strcat(extr_cross, "-o /tmp/trac/cross_data -x 130 -y 75 ");
strcat(extr_cross, "-w 50 -h 50 ");

strcpy(ps_stat, "vstats -i /tmp/trac/temp1_viff ");
strcat(ps_stat, "-f /tmp/trac/statistics_ps -all 0 -mu 0 ");
```

```
strcat(ps_stat, "-var 0 -sd 0 -rms 0 -vmax 0 -xmax 1 ");
strcat(ps_stat, "-ymax 1 -vmin 0 -xmin 0 -ymin 0 -in 0 ");
strcat(ps_stat, "-pin 0 -nin 0 -pts 0 -ppts 0 -npts 0 ");
strcat(ps_stat, "-sk 0 -kur 0 -ent 0 -con 0 -b 1");

strcpy(cs_stat, "vstats -i /tmp/trac/scene1_viff ");
strcat(cs_stat, "-f /tmp/trac/statistics_cs -all 0 -mu 0 ");
strcat(cs_stat, "-var 0 -sd 0 -rms 0 -vmax 0 -xmax 1 ");
strcat(cs_stat, "-ymax 1 -vmin 0 -xmin 0 -ymin 0 -in 0 ");
strcat(cs_stat, "-pin 0 -nin 0 -pts 0 -ppts 0 -npts 0 ");
strcat(cs_stat, "-sk 0 -kur 0 -ent 0 -con 0 -b 1");

strcpy(stat_ps, "vstats -i /tmp/trac/temp1_4mean ");
strcat(stat_ps, "-f /tmp/trac/statistics_ps -all 0 -mu 0 ");
strcat(stat_ps, "-var 0 -sd 0 -rms 0 -vmax 0 -xmax 1 ");
strcat(stat_ps, "-ymax 1 -vmin 0 -xmin 0 -ymin 0 -in 0 ");
strcat(stat_ps, "-pin 0 -nin 0 -pts 0 -ppts 0 -npts 0 ");
strcat(stat_ps, "-sk 0 -kur 0 -ent 0 -con 0 -b 1");

strcpy(stat_cs, "vstats -i /tmp/trac/scene1_4mean ");
strcat(stat_cs, "-f /tmp/trac/statistics_cs -all 0 -mu 0 ");
strcat(stat_cs, "-var 0 -sd 0 -rms 0 -vmax 0 -xmax 1 ");
strcat(stat_cs, "-ymax 1 -vmin 0 -xmin 0 -ymin 0 -in 0 ");
strcat(stat_cs, "-pin 0 -nin 0 -pts 0 -ppts 0 -npts 0 ");
strcat(stat_cs, "-sk 0 -kur 0 -ent 0 -con 0 -b 1");

strcpy(stat_st1, "vstats -i /tmp/trac/sub_temp1 ");
strcat(stat_st1, "-f /tmp/trac/statistics_ps -all 0 -mu 0 ");
strcat(stat_st1, "-var 0 -sd 0 -rms 0 -vmax 0 -xmax 1 ");
strcat(stat_st1, "-ymax 1 -vmin 0 -xmin 0 -ymin 0 -in 0 ");
strcat(stat_st1, "-pin 0 -nin 0 -pts 0 -ppts 0 -npts 0 ");
strcat(stat_st1, "-sk 0 -kur 0 -ent 0 -con 0 -b 1");

strcpy(stat_ss1, "vstats -i /tmp/trac/sub_scene1 ");
strcat(stat_ss1, "-f /tmp/trac/statistics_cs -all 0 -mu 0 ");
strcat(stat_ss1, "-var 0 -sd 0 -rms 0 -vmax 0 -xmax 1 ");
strcat(stat_ss1, "-ymax 1 -vmin 0 -xmin 0 -ymin 0 -in 0 ");
strcat(stat_ss1, "-pin 0 -nin 0 -pts 0 -ppts 0 -npts 0 ");
strcat(stat_ss1, "-sk 0 -kur 0 -ent 0 -con 0 -b 1");

strcpy(stat_s1bland, "vstats -i /tmp/trac/curr_blend_prev ");
strcat(stat_s1bland, "-f /tmp/trac/statistics_cs -all 0 -mu 0 ");
strcat(stat_s1bland, "-var 0 -sd 0 -rms 0 -vmax 0 -xmax 1 ");
strcat(stat_s1bland, "-ymax 1 -vmin 0 -xmin 0 -ymin 0 -in ");
strcat(stat_s1bland, "0 -pin 0 -nin 0 -pts 0 -ppts 0 -npts");
strcat(stat_s1bland, " 0 -sk 0 -kur 0 -ent 0 -con 0 -b 1");

strcpy(stat_t1bland, "vstats -i /tmp/trac/prev_blend_old ");
strcat(stat_t1bland, "-f /tmp/trac/statistics_ps -all 0 -mu 0 ");
strcat(stat_t1bland, "-var 0 -sd 0 -rms 0 -vmax 0 -xmax 1 ");
strcat(stat_t1bland, "-ymax 1 -vmin 0 -xmin 0 -ymin 0 -in ");
strcat(stat_t1bland, "0 -pin 0 -nin 0 -pts 0 -ppts 0 -npts");
strcat(stat_t1bland, " 0 -sk 0 -kur 0 -ent 0 -con 0 -b 1");

strcpy(Cstat, "vstats -i /tmp/trac/real_4stat ");
strcat(Cstat, "-f /tmp/trac/Cstatistics -all 0 -mu 0 ");
strcat(Cstat, "-var 0 -sd 0 -rms 0 -vmax 0 -xmax 1 ");
strcat(Cstat, "-ymax 1 -vmin 0 -xmin 0 -ymin 0 -in ");
strcat(Cstat, "0 -pin 0 -nin 0 -pts 0 -ppts 0 -npts");
strcat(Cstat, " 0 -sk 0 -kur 0 -ent 0 -con 0 -b 1");

strcpy(c2real, "vctor -i /tmp/trac/vtranslate_1 -o /tmp/trac/real_4stat");
strcat(c2real, " -t 1");

strcpy(s1_x_bin, "vmul -i1 /tmp/trac/scene1_viff ");
```

```
strcat(s1_x_bin, "-i2 /tmp/trac/tmp_scene1 ");
strcat(s1_x_bin, "-o /tmp/trac/scene1_viff");

strcpy(t1_x_bin, "vmul -i1 /tmp/trac/temp1_viff ");
strcat(t1_x_bin, "-i2 /tmp/trac/tmp_temp1 ");
strcat(t1_x_bin, "-o /tmp/trac/temp1_viff");

strcpy(blend_s1, "vblend -i1 /tmp/trac/scene1_viff -i2 ");
strcat(blend_s1, "/tmp/trac/temp1_viff -x 0.8 -o ");
strcat(blend_s1, "/tmp/trac/curr_blend_prev");

strcpy(blend_t1, "vblend -i1 /tmp/trac/temp1_viff -i2 ");
strcat(blend_t1, "/tmp/trac/old_viff -x 0.8 -o ");
strcat(blend_t1, "/tmp/trac/prev_blend_old");

strcpy(blend_x_t1, "vmul -i1 /tmp/trac/temp1_viff -i2 ");
strcat(blend_x_t1, "/tmp/trac/tmp_temp1 -o /tmp/trac/temp1_viff");

strcpy(blend_x_s1, "vmul -i1 /tmp/trac/scene1_viff -i2 ");
strcat(blend_x_s1, "/tmp/trac/tmp_scene1 -o /tmp/trac/scene1_viff");

strcpy(cmplx_t1, "vrtoc -i  tmp/trac/prev_blend_old -o ");
strcat(cmplx_t1, "/tmp/trac/blend_2cmplx_t1 -t 1");

strcpy(cmplx_s1, "vrtoc -i1 /tmp/trac/curr_blend_prev -o ");
strcat(cmplx_s1, "/tmp/trac/blend_2cmplx_s1 -t 1");

strcpy(hpf_t1, "vhpf -i /tmp/trac/blend_2cmplx_t1 -o ");
strcat(hpf_t1, "/tmp/trac/prev_blend_old -n 3 -c 0.5");

strcpy(hpf_s1, "vhpf -i /tmp/trac/blend_2cmplx_s1 -o ");
strcat(hpf_s1, "/tmp/trac/curr_blend_prev -n 3 -c 0.5");

strcpy(t1_2real, "vctor -i /tmp/trac/sub_temp1 -o ");
strcat(t1_2real, "/tmp/trac/t1_for_stat -t 1");

strcpy(s1_2real, "vctor -i /tmp/trac/sub_scene1 -o ");
strcat(s1_2real, "/tmp/trac/s1_for_stat -t 1");

strcpy(a_2asc1, "vprdata -i /tmp/trac/atarg_4stat ");
strcat(a_2asc1, "-f /tmp/trac/ttarg_ascii_data -m 0");

strcpy(b_2asc1, "vprdata -i /tmp/trac/btarg_4stat ");
strcat(b_2asc1, "-f /tmp/trac/starg_ascii_data -m 0");

strcpy(a_2asc2, "vprdata -i /tmp/trac/a_targ_rect ");
strcat(a_2asc2, "-f /tmp/trac/a_targ_ascii -m 0");

strcpy(b_2asc2, "vprdata -i /tmp/trac/b_targ_rect ");
strcat(b_2asc2, "-f /tmp/trac/b_targ_ascii -m 0");

strcpy(a_2asc3, "vprdata -i /tmp/trac/tbkgnd_4stat ");
strcat(a_2asc3, "-f /tmp/trac/tbkgnd_ascii_data -m 0");

strcpy(b_2asc3, "vprdata -i /tmp/trac/sbkgnd_4stat ");
strcat(b_2asc3, "-f /tmp/trac/sbkgnd_ascii_data -m 0");

strcpy(a_2byte1, "vconvert -i /tmp/trac/a_targ_2viff ");
strcat(a_2byte1, "-o /tmp/trac/a_targ_2byte -n 99 -t byte -b 0");

strcpy(b_2byte1, "vconvert -i /tmp/trac/b_targ_2viff ");
strcat(b_2byte1, "-o /tmp/trac/b_targ_2byte -n 99 -t byte -b 0");

strcpy(a_2byte2, "vconvert -i /tmp/trac/a_targ_2viff ");
strcat(a_2byte2, "-o /tmp/trac/a_targ_2byte -n 255 -t byte -b 0");
```

```
strcpy(b_2byte2, "vconvert -i /tmp/trac/b_targ_2viff ");
strcat(b_2byte2, "-o /tmp/trac/b_targ_2byte -n 255 -t byte -b 0");

strcpy(dyn1_s1, "vdyth -i /tmp/trac/curr_blend_prev ");
strcat(dyn1_s1, "-o /tmp/trac/bin_3x3_s1 -w 3 -h 3 -v ");
strcat(dyn1_s1, "255 -t 0");

strcpy(dyn1_t1, "vdyth -i /tmp/trac/prev_blend_old ");
strcat(dyn1_t1, "-o /tmp/trac/bin_3x3_t1 -w 3 -h 3 -v ");
strcat(dyn1_t1, "255 -t 0");

strcpy(dyn2_s1, "vdyth -i /tmp/trac/scene1_viff ");
strcat(dyn2_s1, "-o /tmp/trac/bin_3x3_s1 -w 3 -h 3 -v ");
strcat(dyn2_s1, "255 -t 0");

strcpy(dyn2_t1, "vdyth -i /tmp/trac/temp1_viff ");
strcat(dyn2_t1, "-o /tmp/trac/bin_3x3_t1 -w 3 -h 3 -v ");
strcat(dyn2_t1, "255 -t 0");

strcpy(and_s1, "vand -i1 /tmp/trac/bin_4and_s1 ");
strcat(and_s1, "-i2 /tmp/trac/bin_3x3_s1 -o ");
strcat(and_s1, "/tmp/trac/tmp_scene1");

strcpy(and_t1, "vand -i1 /tmp/trac/bin_4and_t1 ");
strcat(and_t1, "-i2 /tmp/trac/bin_3x3_t1 -o ");
strcat(and_t1, "/tmp/trac/tmp_temp1");


strcpy(padt1, "vpad -i /tmp/trac/sub_temp1 ");
strcat(padt1, "-o /tmp/trac/T1_disp -r 64 -c 64 ");
strcat(padt1, "-d 0 -e 0 -j 255 -k 0");



/*-----------------------------------------*/
/* Initialize miscellaneous command strings. */
/*-----------------------------------------*/
strcpy(mean_cs, "grep Mean /tmp/trac/statistics_cs ");
strcat(mean_cs, "> /tmp/trac/cs_mean_data");

strcpy(mean_ps, "grep Mean /tmp/trac/statistics_ps ");
strcat(mean_ps, "> /tmp/trac/ps_mean_data");

strcpy(corr_mean, "grep Mean /tmp/trac/Cstatistics ");
strcat(corr_mean, "> /tmp/trac/mean_data");

strcpy(corr_var, "grep Variance /tmp/trac/Cstatistics ");
strcat(corr_var, "> /tmp/trac/var_data");

strcpy(corr_stdev, "grep Std. /tmp/trac/Cstatistics ");
strcat(corr_stdev, "> /tmp/trac/stdev_data");

strcpy(corr_max, "grep High /tmp/trac/Cstatistics ");
strcat(corr_max, "> /tmp/trac/max_data");

strcpy(corr_min, "grep Low /tmp/trac/Cstatistics ");
strcat(corr_min, "> /tmp/trac/min_data");

strcpy(copy_cs, "cp /tmp/trac/scene1_viff /tmp/trac/sviff_4stat");

strcpy(copy_ps, "cp /tmp/trac/temp1_viff /tmp/trac/tviff_4stat");

strcpy(t1_4stat, "cp /tmp/trac/tmp_temp1 /tmp/trac/temp1_4stat");
```

```
strcpy(s1_4stat, "cp /tmp/trac/tmp_scene1 /tmp/trac/scene1_4stat");

strcpy(gt1_4stat, "cp /tmp/trac/trev_bin_4stat /tmp/trac/asecond_4stat");

strcpy(hi_cs, "grep High /tmp/trac/statistics_cs ");
strcat(hi_cs, "> /tmp/trac/high_data");

strcpy(hi_ps, "grep High /tmp/trac/statistics_ps ");
strcat(hi_ps, "> /tmp/trac/high_data");

strcpy(gs1_4stat, "cp /tmp/trac/srev_bin_4stat /tmp/trac/bsecond_4stat");

strcpy(ct1_4stat, "cp /tmp/trac/temp1_viff /tmp/trac/temp1_4stat");

strcpy(cs1_4stat, "cp /tmp/trac/scene1_viff /tmp/trac/scene1_4stat");

strcpy(clr_scr, "clear");

/*-------------------------------------------*/
/* Copy next image file to /tmp/trac/current_scene. */
/*-------------------------------------------*/
while(mode=='f')
{
  if((fp5 = fopen(files[count], "rb"))==NULL)
  {
    printf("Error opening %s!\n", files[count]);
    exit(1);
  }
  if((fp1 = fopen("/tmp/trac/current_scene", "wb"))==NULL)
  {
    printf("Error opening /tmp/trac/current_scene!\n");
    exit(1);
  }
  character = fgetc(fp5);
  if(ferror(fp5))
  {
    printf("Error reading %s!\n", files[count]);
    exit(1);
  }
  while(character!=EOF)
  {
    fputc(character, fp1);
    if(ferror(fp1))
    {
      printf("Error writing to /tmp/trac/current_scene!\n");
      exit(1);
    }
    character = fgetc(fp5);
    if(ferror(fp5))
    {
      printf("Error reading %s!\n", files[count]);
      exit(1);
    }
  }
  fclose(fp5);
  fclose(fp1);

  /*-------------------------------------------------*/
  /* Display an image in its original form and rename */
  /* the file current_scene to previous_scene.        */
  /*-------------------------------------------------*/
  system(r2v_cs);
  system(pad3_2x5);
  system(delay);
  system(byte_fov);
```

```
if(pass==0)
{
  if(display=='y')
  {
    system(display_fov);
  }
  swap=rename("/tmp/trac/current_scene", "/tmp/trac/previous_scene");
  if(swap!=0)
  {
    printf("File renaming operation 85 failed!\n");
    exit(1);
  }
}
if(pass!=0)
{
  swap = rename("/tmp/trac/previous_scene", "/tmp/trac/old_scene");
  if(swap!=0)
  {
    printf("File renaming operation 86 failed!\n");
    exit(1);
  }
  swap=rename("/tmp/trac/current_scene", "/tmp/trac/previous_scene");
  if(swap!=0)
  {
    printf("File renaming operation 87 failed!\n");
    exit(1);
  }
}

/*-----------------------------------------------*/
/* Determine target range and display to screen. */
/*-----------------------------------------------*/
for(i=0; i<128; i++)  string[i] = files[count][i];
length = strlen(string);
for(j=0; j<5; j++)  term1[j] = string[length - 9 + j];
term1[j] = '\0';
fr_info = atoi(term1);

if(index==0)
{
  printf("\n");
  system(clr_scr);
  printf("Target\nRange\n");
  printf("------\n");
  index = 20;
}
printf("%5d\n", fr_info);
index = index - 1;

count = count + 1;

/*---------------------------------------*/
/* Trigger the display of targeting menu. */
/*---------------------------------------*/
if(pass==2)
{
  if((fp2 = fopen("/tmp/trac/start1_flag", "w"))==NULL)
  {
    printf("Error opening /tmp/trac/start1_flag!\n");
    exit(1);
  }
  fputc(on, fp2);
  fclose(fp2);
}
```

```
pass = pass + 1;
if((fp4 = fopen("/tmp/trac/state_flag", "r"))==NULL)
{
  printf("Error opening /tmp/trac/state_flag\n!");
  exit(1);
}
state_flag = fgetc(fp4);
fclose(fp4);

/*-------------------------------------------*/
/* Determine targeting state requested by user */
/* and reset the file state_flag.              */
/*-------------------------------------------*/
if(state_flag==on)
{
  state_flag = off;
  if((fp5 = fopen("/tmp/trac/state_flag", "r"))==NULL)
  {
    printf("Error opening /tmp/trac/state_flag!\n");
    exit(1);
  }
  fputc(off, fp5);
  fclose(fp5);

  if((fp1 = fopen("/tmp/trac/state_info", "r"))==NULL)
  {
    printf("Error opening /tmp/trac/state_info\n!");
    exit(1);
  }
  state = fgetc(fp1);

  /*-----------------------------------------------*/
  /* Determine the number of image frames to skip. */
  /*-----------------------------------------------*/
  if(state=='s')
  {
    if((fp2 = fopen("/tmp/trac/start1_flag", "w"))==NULL)
    {
      printf("Error opening /tmp/trac/start1_flag!\n");
      exit(1);
    }
    fputc(off, fp2);
    fclose(fp2);

    fgets(string, 128, fp1);
    fgets(string, 128, fp1);
    step = atoi(string);
    count = count + step;
    if((argc - count)<3)  state = 'e';
    job3 = 1;
    pass = 1;
  }
  fclose(fp1);
}

/*-----------------------------------------------------*/
/* If last image in sequence has been displayed or per  */
/* user's request, kill background process, and exit.   */
/*-----------------------------------------------------*/
if(state=='e' || (argc-count)==1)
{
  strcpy(search, "ps -aux | grep putimage | grep -v");
  strcat(search, " 'grep' > /tmp/trac/job_numbers");
  system(search);
```

```
    if((fp4 = fopen("/tmp/trac/job_numbers", "r"))==NULL)
    {
      printf("Error opening /tmp/trac/job_numbers!\n");
      exit(1);
    }

    fgets(string, 128, fp4);
    for(m=0; m<6; m++)   ps_value[m] = string[m + 9];
    ps_value[m] = '\0';
    job1 = atoi(ps_value);
    fclose(fp4);

    sprintf(k_displ_fov, "kill %d", job1);
    system(k_displ_fov);
    if((fp5 = fopen("/tmp/trac/EOS_flag", "w"))==NULL)
    {
      printf("Error opening /tmp/trac/EOS_flag!\n");
      exit(1);
    }
    fputc(on, fp5);
    fclose(fp5);

    printf("%c\n", '\007');

    if(job3==0)
    {
      printf("/*---------------------------------------");
      printf("--------*/\n");
      printf("/* This program has been terminated either");
      printf(" in re- */\n");
      printf("/* ponse to the user's request or because ");
      printf("the di- */\n");
      printf("/* gitized images for this sequence have b");
      printf("een ex- */\n");
      printf("/* hausted. If the program 'trac_menu' has");
      printf(" not    */\n");
      printf("/* already been terminated, please reponse");
      printf(" with   */\n");
      printf("/* option 'e' to terminate it now.        ");
      printf("        */\n");
      printf("/*---------------------------------------");
      printf("--------*/\n");
    }
    else if(job3==1)
    {
      printf("/*---------------------------------------");
      printf("--------*/\n");
      printf("/* Program terminated. The requested numbe");
      printf("r of    */\n");
      printf("/* image frames to be skipped exceeded the");
      printf(" number */\n");
      printf("/* of remaining un-displayed image frames ");
      printf(" in the */\n");
      printf("/* seqeuence.  Please reponse with option ");
      printf(" 'e' to */\n");
      printf("/* terminate the program 'trac_menu' now. ");
      printf("       */\n");
      printf("/*---------------------------------------");
      printf("--------*/\n");
    }
    exit(0);
}

/*-----------------------*/
```


121

```
/* Target tracking loop. */
/*-----------------------*/
while(state=='t')
{
  if(cycle==0)
  {
    index = 0;
    if((fp1 = fopen("/tmp/trac/coords_flag_1", "r"))==NULL)
    {
      puts("Error reading from /tmp/trac/coords_flag_1!");
      exit(1);
    }
    /*---------------------------------*/
    /* Delay until user has inputted   */
    /* required correlation parameters. */
    /*---------------------------------*/
    temp = fgetc(fp1);
    while(temp!='on')
    {
      fclose(fp1);
      if((fp1 = fopen("/tmp/trac/coords_flag_1", "r"))==NULL)
      {
        puts("Error reading from /tmp/trac/coords_flag_1!");
        exit(1);
      }
      temp = fgetc(fp1);
    }
    fclose(fp1);

    if((fp2 = fopen("/tmp/trac/coords_flag_1", "w"))==NULL)
    {
      puts("Error writing to /tmp/trac/coords_flag_1!");
      exit(1);
    }
    fputc(off, fp2);
    fclose(fp2);

    /*-------------------------------------------------*/
    /* Retrieve inputted target coordinates and options. */
    /*-------------------------------------------------*/
    if((fp4 = fopen("/tmp/trac/coords_info_1", "r"))==NULL)
    {
      puts("Error reading from /tmp/trac/coords_info_1!\n");
      exit(1);
    }
    fgets(string, 128, fp4);
    x_value = atoi(string);
    fgets(string, 128, fp4);
    y_value = atoi(string);
    fgets(string, 128, fp4);
    mthd = string[0];

    /*-------------------------------------*/
    /* Determine correlation method chosen. */
    /*-------------------------------------*/
    if(mthd=='a')
    {
      strcpy(str0, "Thresholded (0 or 255)");
      strcpy(rename0, "a_");
    }
    else if(mthd=='b')
    {
      strcpy(str0, "Thresholded (w/ gray-scale)");
      strcpy(rename0, "b_");
    }
```

```
else if(mthd=='c')
{
  strcpy(str0, "Truth template and scene");
  strcpy(rename0, "c_");
}
else if(mthd=='d')
{
  strcpy(str0, "Truth templ w/ unmodified scene");
  strcpy(rename0, "d_");
}
else if(mthd=='e')
{
  strcpy(str0, "Thresholded blend (0 or 255)");
  strcpy(rename0, "e_");
}
else if(mthd=='f')
{
  strcpy(str0, "Thresholded blend (w/ gray-scale)");
  strcpy(rename0, "f_");
}
else if(mthd=='g')
{
  strcpy(str0, "Unmodified gray-scale images");
  strcpy(rename0, "g_");
}
else
{
  strcpy(str0, "Optical adaptive tracking simulation.");
  strcpy(rename0, "zo_");
}

rename3[0] = '.';
rename3[1] = '0';
rename3[2] = '0';
rename3[3] = '\0';

fgets(string, 128, fp4);
if(mthd=='a' || mthd=='b' || mthd=='e' || mthd=='f')
{
  factor = atof(string);
  rename3[0] = string[1];
  rename3[1] = string[2];
  rename3[2] = string[3];
  rename3[3] = '\0';
}
fgets(string, 128, fp4);
size = string[0];

/*----------------------------------------*/
/* Determine binarization technique chosen. */
/*----------------------------------------*/
strcpy(rename1, "none_");
if(mthd=='a' || mthd=='b' || mthd=='e' || mthd=='f')
{
  fgets(string, 128, fp4);
  bin_state = string[0];
  strcpy(str2, "Scene average");
  strcpy(rename1, "scen_");

  if(bin_state=='c')
  {
    strcpy(str2, "Cline segmentation");
    strcpy(rename1, "clin_");
  }
```

```
  if(bin_state=='g')
  {
    bin_state = 'a';
    strcpy(str2, "Line-by-line average");
    strcpy(rename1, "line_");
  }
}

/*--------------------------*/
/* Determine if edger used. */
/*--------------------------*/
if(mthd=='g')
{
  fgets(string, 128, fp4);
  fgets(string, 128, fp4);
  edger = string[0];
  if(edger=='y')
  {
    strcpy(str2, "Edge Extraction");
    strcpy(rename1, "edge_");
  }
}


fclose(fp4);

/*-----------------------------------*/
/* Set the template and scene sizes. */
/*-----------------------------------*/
if(size=='l')
{
  y_template1_size = 40;
  x_template1_size = 64;
  y_scene1_size = 120;
  x_scene1_size = 192;
  pad_size = 256;
  strcpy(str1, "Large (Tmpl=64x40, Scn=192x120)");
  strcpy(rename2, "l");
}
else if(size=='s')
{
  y_template1_size = 20;
  x_template1_size = 30;
  y_scene1_size = 38;
  x_scene1_size = 60;
  pad_size = 64;
  strcpy(str1, "Small (Tmpl=30x20, Scn=60x38)");
  strcpy(rename2, "s");
}
else if(size=='m')
{
  y_template1_size = 28;
  x_template1_size = 42;
  y_scene1_size = 72;
  x_scene1_size = 116;
  pad_size = 128;
  strcpy(str1, "Medium (Tmpl=42x28, Scn=116x72)");
  strcpy(rename2, "m");
}
else if(size=='a')
{
  y_template1_size = 20;
  x_template1_size = 30;
  y_scene1_size = 38;
  x_scene1_size = 60;
```

```
  pad_size = 64;
  strcpy(str1, "Adapt/S (Tmpl=Adapt, Scn=60x38)");
  strcpy(rename2, "s");
}
else
{
  y_template1_size = 20;
  x_template1_size = 30;
  y_scene1_size = 320;
  x_scene1_size = 499;
  pad_size = 512;
  strcpy(str1, "Adapt/F (Tmpl=Adapt, Scn=499x320)");
  strcpy(rename2, "f");
}


for(m=0; m<6; m++)
{
  term2[m] = tru_cntr[count - 1][m];
}
term2[m] = '\0';
tr_info = atoi(term2);
for(i=0; i<3; i++)
{
  /*-----------------------------*/
  /* Retrieve the x- coordinate */
  /* of target locations.       */
  /*-----------------------------*/
  j = 0;
  temp = tru_cntr[count - 1][m];
  while(temp!=',')
  {
    term1[j] = temp;
    j = j + 1;
    temp = tru_cntr[count - 1][m + j];
  }
  term1[j] = '\0';
  x_term = atoi(term1);
  xf_term = atof(term1);
  j = j + 1;

  /*-----------------------------*/
  /* Retrieve the y- coordinate */
  /* of target locations.       */
  /*-----------------------------*/
  for(k=0; k<7; k++)
  {
    term2[k] = tru_cntr[count - 1][m + j + k];
  }
  term2[k] = '\0';
  y_term = atoi(term2);
  y_term = 512 - y_term;
  yf_term = atof(term2);
  yf_term = 512.0 - yf_term;
  m = m + j + k + 21;

  integ1 = ((x_value - x_term) * (x_value - x_term));
  integ2 = ((y_value - y_term) * (y_value - y_term));
  result = sqrt((double) integ1 + (double) integ2);

  if(i==0)
  {
    distance = result;
    x_coord = x_term;
    y_coord = y_term;
```

```
    xf_coord = xf_term;
    yf_coord = yf_term;
    targ_num = i + 1;
  }

  /*-----------------------------------------------*/
  /* Determine whether inputted target coordinates */
  /* correspond to an actual target location.      */
  /*-----------------------------------------------*/
  else if((distance - result)>0)
  {
    distance = result;
    x_coord = x_term;
    y_coord = y_term;
    xf_coord = xf_term;
    yf_coord = yf_term;
    targ_num = i + 1;
  }
}
if((distance - 20.0)>0)
{
  x_coord = x_value;
  y_coord = y_value;
  xf_coord = ((double) x_value);
  yf_coord = ((double) y_value);
  targ_num = 0;
}

if(targ_num==0)
{
  strcpy(end, "_0");
}
else if(targ_num==1)
{
  strcpy(end, "_1");
}
else if(targ_num==2)
{
  strcpy(end, "_2");
}
else strcpy(end, "_3");

/*---------------------------------*/
/* Pass true target coordinates to */
/* the program trac_menu.          */
/*---------------------- ----------*/
if((fp5 = fopen("/tmp/trac/coords_info_2", "w"))==NULL)
{
  printf("Error opening /tmp/trac/coords_info_2!\n");
  exit(1);
}
fprintf(fp5, "%d\n%6.2f\n", targ_num, xf_coord);
fprintf(fp5, "%6.2f\n%d\n", yf_coord, tr_info);
fclose(fp5);

if((fp1 = fopen("/tmp/trac/coords_flag_2", "w"))==NULL)
{
  printf("Error opening /tmp/trac/coords_flag_2!\n");
  exit(1);
}
fputc(on, fp1);
fclose(fp1);

/*-----------------------------------------------*/
/* Wait for user's response to allow tracking  */
```

```
/* on true versus inputted target coordinates. */
/*-----------------------------------------------*/
if(targ_num!=0)
{
  if((fp2 = fopen("/tmp/trac/coords_flag_1", "r"))==NULL)
  {
    printf("Error opening /tmp/trac/coords_flag_1!\n");
    exit(1);
  }

  temp = fgetc(fp2);
  while(temp!=on)
  {
    fclose(fp2);
    if((fp2 = fopen("/tmp/trac/coords_flag_1", "r"))==NULL)
    {
      printf("Error opening /tmp/trac/coords_flag_1!\n");
      exit(1);
    }
    temp = fgetc(fp2);
  }
  fclose(fp2);

  if((fp4 = fopen("/tmp/trac/coords_flag_1", "w"))==NULL)
  {
    printf("Error opening /tmp/trac/coords_flag_1!\n");
    exit(1);
  }
  fputc(off, fp4);
  fclose(fp4);

  /*--------------------------------*/
  /* Retrieve the user's response. */
  /*--------------------------------*/
  if((fp5 = fopen("/tmp/trac/coords_info_1", "r"))==NULL)
  {
    puts("Error reading from /tmp/trac/coords_info_1!\n");
    exit(1);
  }
  temp = fgetc(fp5);
  fclose(fp5);

  if(temp=='n')
  {
    x_coord = x_value;
    y_coord = y_value;
    xf_coord = ((double) x_value);
    yf_coord = ((double) y_value);
  }
}

x_coord = 150 + x_coord;
y_coord = 40 + y_coord;

xf_cntr = xf_coord;
yf_cntr = yf_coord;

xf_coord = 150.0 + xf_coord;
yf_coord = 40.0 + yf_coord;

/*-----------------------------------------------*/
/* Generate adaptive template dimensions and     */
/* location for correlation.                     */
/*-----------------------------------------------*/
if(size=='a' || size=='f')
```

```
{
  xedge = x_coord - 200;
  yedge = y_coord - 90;
  if(xedge < 0)
  {
    xedge = 0;
  }
  if(xedge > 398)
  {
    xedge = 398;
  }
  if(yedge < 0)
  {
    yedge = 0;
  }
  if(yedge > 220)
  {
    yedge = 220;
  }
  strcpy(extr_win, "vextract -i /tmp/trac/scene1_viff -o - ");
  sprintf(string, "-x %d -y %d -w 100 -h 100", xedge, yedge);
  strcat(extr_win, string);
  strcat(extr_win, " | viff2mat -i - ");
  strcat(extr_win, "-o /tmp/trac/tmp_1 -j 1 -b 0 ");
  strcat(extr_win, "-d 0 -f 0 ");

  system(extr_win);


  if((fp1 = fopen("/tmp/trac/tmp_1", "r"))==NULL)
  {
    printf("Error opening /tmp/trac/tmp_1!\n");
    exit(1);
  }

  fgets(string, 128, fp1);

  for (yy = 0; yy <= 99; ++yy) {
    for (xx = 0; xx <= 99; ++xx) {

      if((fscanf(fp1, "%d", &xy[xx][yy]))==NULL)
      {
        printf("Error reading from /tmp/trac/tmp_1!\n");
        exit(1);
      }
    }
    fscanf(fp1, "%*s");
  }
  fclose(fp1);

  tmpx = x_coord - 150 - xedge;
  tmpy = y_coord - 40 - yedge;

  /*-----------------------------------------------*/
  /* Check to see if current selected pixel is part */
  /* of a target.                                   */
  /*-----------------------------------------------*/

  if(xy[tmpx][tmpy] <= 70)
  {
    count = 0;
    a1 = tmpy;
    b1 = tmpx;
    c1 = tmpy;
    d1 = tmpx;
```

```
while(count == 0)
{
  a1 = a1 - 1;
  b1 = b1 + 1;
  c1 = c1 + 1;
  d1 = d1 - 1;
  if(a1 < 0) a1 = 0;
  if(b1 > 100) b1 = 100;
  if(c1 > 100) c1 = 100;
  if(d1 < 0) d1 = 0;


  for(xx = d1; xx <= b1; ++xx) {
    if(xy[xx][a1] >= 71)
    {
      count = 1;
      if(xy[xx][a1] >= (xy[tmpx][tmpy] + 1))
      {
        tmpx = xx;
        tmpy = a1;
      }
    }
  }

  for(yy = a1; yy <= c1; ++yy) {
    if(xy[b1][yy] >= 71)
    {
      count = 1;
      if(xy[b1][yy] >= (xy[tmpx][tmpy] + 1))
      {
        tmpx = b1;
        tmpy = yy;
      }
    }
  }

  for(xx = d1; xx <= b1; ++xx) {
    if(xy[xx][c1] >= 71)
    {
      count = 1;
      if(xy[xx][c1] >= (xy[tmpx][tmpy] + 1))
      {
        tmpx = xx;
        tmpy = c1;
      }
    }
  }

  for(yy = a1; yy <= c1; ++yy) {
    if(xy[d1][yy] >= 71)
    {
      count = 1;
      if(xy[d1][yy] >= (xy[tmpx][tmpy] + 1))
      {
        tmpx = d1;
        tmpy = yy;
      }
    }
  }
  if((a1==0)&&(b1==99)&&(c1==99)&&(d1==0))
  {
    tmpx = permx;
    tmpy = permy;
  }
```


10

```
  }
}

/*-------------------------------------*/
/* Find bounding box of target.        */
/*-------------------------------------*/

a1 = tmpy - 1;
a2 = 0;
b1 = tmpx + 1;
b2 = 0;
c1 = tmpy + 1;
c2 = 0;
d1 = tmpx - 1;
d2 = 0;

while(a2==0 || b2==0 || c2==0 || d2==0)
{
  if(a2 == 0)
  {
    a1 = a1 - 1;
  }
  a2 = 0;
  count = 0;
  if(a1 < 0)
  {
    a1 = 0;
    a2 = 1;
  }
  for(xx = d1; xx <= b1; ++xx) {
    if(xy[xx][a1] <= 70)
    {
      count = count + 1;
    }
  }
  check = (count / (b1 - d1));
  if(check >= 0.9)
  {
    a2 = 1;
  }

  if(b2 == 0)
  {
    b1 = b1 + 1;
  }
  b2 = 0;
  count = 0;
  if(b1 > 100)
  {
    b1 = 100;
    b2 = 1;
  }
  for(yy = a1; yy <= c1; ++yy) {
    if(xy[b1][yy] <= 70)
    {
      count = count + 1;
    }
  }
  check = (count / (c1 - a1));
  if(check >= 0.9)
  {
    b2 = 1;
  }

  if(c2 == 0)
```

```
  {
    c1 = c1 + 1;
  }
  c2 = 0;
  count = 0;
  if(c1 > 100)
  {
    c1 = 100;
    c2 = 1;
  }
  for(xx = d1; xx <= b1; ++xx) {
    if(xy[xx][c1] <= 70)
    {
      count = count + 1;
    }
  }
  check = (count / (b1 - d1));
  if(check >= 0.9)
  {
    c2 = 1;
  }

  if(d2 == 0)
  {
    d1 = d1 - 1;
  }
  d2 = 0;
  count = 0;
  if(d1 < 0)
  {
    d1 = 0;
    d2 = 1;
  }
  for(yy = a1; yy <= c1; ++yy) {
    if(xy[d1][yy] <= 70)
    {
      count = count + 1;
    }
  }
  check = (count / (c1 - a1));
  if(check >= 0.9)
  {
    d2 = 1;
  }

}

x_coord = ((b1 + d1) / 2) + xedge;
y_coord = ((c1 + a1) / 2) + yedge;
perx = x_coord;
pery = y_coord;

x_template1_size = b1 - d1;
y_template1_size = c1 - a1;
if(x_template1_size > 64)
{
  x_template1_size = 64;
}
if(y_template1_size > 64)
{
  y_template1_size = 64;
}

x_value = x_coord;
y_value = y_coord;
```

```
  xf_coord = ((double) (bi + di) / 2) + xedge;
  yf_coord = ((double) (ci + ai) / 2) + yedge;
  xf_cntr = xf_coord;
  yf_cntr = yf_coord;

  x_coord = 150 + x_coord;
  y_coord = 40 + y_coord;
  xf_coord = 150.0 + xf_coord;
  yf_coord = 40.0 + yf_coord;

}


/*-------------------------------------------*/
/* Compute upper left x- and y- pixel coordinates */
/* for sub-images template1 and scene1.          */
/*-------------------------------------------*/
x_template1 = x_coord - (x_template1_size / 2);
y_template1 = y_coord - (y_template1_size / 2);
x_scene1 = x_coord - (x_scene1_size / 2);
y_scene1 = y_coord -( y_scene1_size / 2);
if(size=='f')
{
  x_scene1 = 150;
  y_scene1 = 40;
}

if(edger!='y')
{
  strcpy(extr_s1, "vextract -i /tmp/trac/scene1_pad_4x8 ");
  strcat(extr_s1, "-o /tmp/trac/sub_scene1 ");
  sprintf(string, "-x %d -y %d ", x_scene1, y_scene1);
  strcat(extr_s1, string);
  sprintf(string, "-w %d ", x_scene1_size);
  strcat(extr_s1, string);
  sprintf(string, "-h %d", y_scene1_size);
  strcat(extr_s1, string);

  strcpy(extr_t1, "vextract -i /tmp/trac/temp1_pad_4x8 ");
  strcat(extr_t1, "-o /tmp/trac/sub_temp1 ");
  sprintf(string, "-x %d ", x_template1);
  strcat(extr_t1, string);
  sprintf(string, "-y %d ", y_template1);
  strcat(extr_t1, string);
  sprintf(string, "-w %d ", x_template1_size);
  strcat(extr_t1, string);
  sprintf(string, "-h %d", y_template1_size);
  strcat(extr_t1, string);
}

if(edger=='y')
{
  strcpy(extr_s1, "vdrf -i /tmp/trac/scene1_pad_4x8 -o - ");
  strcat(extr_s1, "-a1 0.35 -a2 0.35 -w 9 -t1 6 -t2 8 -l 7 ");
  strcat(extr_s1, " | vextract -i - ");
  strcat(extr_s1, "-o /tmp/trac/sub_scene1 ");
  sprintf(string, "-x %d -y %d ", x_scene1, y_scene1);
  strcat(extr_s1, string);
  sprintf(string, "-w %d ", x_scene1_size);
  strcat(extr_s1, string);
  sprintf(string, "-h %d", y_scene1_size);
  strcat(extr_s1, string);

  strcpy(extr_t1, "vdrf -i /tmp/trac/temp1_pad_4x8 -o - ");
  strcat(extr_t1, "-a1 0.35 -a2 0.35 -w 9 -t1 6 -t2 8 -l 7 ");
```

```
  strcat(extr_t1, " | vextract -i - ");
  strcat(extr_t1, "-o /tmp/trac/sub_temp1 ");
  sprintf(string, "-x %d ", x_template1);
  strcat(extr_t1, string);
  sprintf(string, "-y %d ", y_template1);
  strcat(extr_t1, string);
  sprintf(string, "-w %d ", x_template1_size);
  strcat(extr_t1, string);
  sprintf(string, "-h %d", y_template1_size);
  strcat(extr_t1, string);

}


/*-----------------------------------------*/
/* Date and time stamp the output file used */
/* to record the correlation process.       */
/*-----------------------------------------*/
if(targ_num!=0)
{
  strcpy(summary, name1);
  strcat(summary, rename0);
  strcat(summary, rename1);
  strcat(summary, rename2);
  strcat(summary, rename3);

  sprintf(rename4, "mkdir %s", summary);
  system(rename4);

  strcat(summary, "/");
  strcpy(plotname, summary);
  if(targ_num==1)
  {
    strcat(plotname, "range_error.1");
  }
  else if(targ_num==2)
  {
    strcat(plotname, "range_error.2");
  }
  else
  {
    strcat(plotname, "range_error.3");
  }

  if((fp6 = fopen(plotname, "w"))==NULL)
  {
    printf("Error opening file range_error!");
    exit(1);
  }


  strcpy(plotname, summary);
  if(targ_num==1)
  {
    strcat(plotname, "range_location.1");
  }
  else if(targ_num==2)
  {
    strcat(plotname, "range_location.2");
  }
  else
  {
    strcat(plotname, "range_location.3");
  }
```

```
    if((fp7 = fopen(plotname, "w"))==NULL)
    {
      printf("Error opening file range_location!");
      exit(1);
    }


    strcpy(plotname, summary);
    if(targ_num==1)
    {
      strcat(plotname, "other_targets.1");
    }
    else if(targ_num==2)
    {
      strcat(plotname, "other_targets.2");
    }
    else
    {
      strcat(plotname, "other_targets.3");
    }

    if((fp8 = fopen(plotname, "w"))==NULL)
    {
      printf("Error opening file other_targets!");
      exit(1);
    }

    strcpy(plotname, summary);

    strcat(summary, rename0);
    strcat(summary, rename1);
    strcat(summary, rename2);
    strcat(summary, rename3);

    strcat(summary, end);

    if((fp3 = fopen(summary, "w"))==NULL)
    {
      printf("Error opening %s!\n", summary);
      exit(1);
    }

    printf("%c\n", '\007');
    printf("*** Correlation summary is being recorded");
    printf(" in: %s ***\n", summary);
  }

  digit1 = 0;
  digit2 = 0;
  digit3 = 1;
}

if(opt=='s' || opt=='m' || opt=='l' || opt=='t')
{
  cycle = 0;
  opt = 'a';
}

old_x_temp1 = x_template1;
old_y_temp1 = y_template1;
old_x_scene1 = x_scene1;
old_y_scene1 = y_scene1;

/*-------------------------------------------------*/
```

```
/* Retrieve next image and copy to 'current_scene'. */
/*--------------------------------------------------*/
if((fp1 = fopen(files[count], "rb"))==NULL)
{
  printf("Error opening %s!\n", files[count]);
  exit(1);
}
if((fp2 = fopen("/tmp/trac/current_scene", "wb"))==NULL)
{
  printf("Error opening 1 /tmp/trac/current_scene!\n");
  exit(1);
}
character = fgetc(fp1);
if(ferror(fp1))
{
  printf("Error reading %s!\n", files[count]);
  exit(1);
}
while(character!=EOF)
{
  fputc(character, fp2);
  if(ferror(fp2))
  {
    printf("Error writing to /tmp/trac/current_scene!\n");
    exit(1);
  }
  character = fgetc(fp1);
  if(ferror(fp1))
  {
    printf("Error reading %s!\n", files[count]);
    exit(1);
  }
}
fclose(fp1);
fclose(fp2);

/*-----------------------------------*/
/* Convert image data to VIFF format. */
/*-----------------------------------*/
system(r2v_cs);
system(r2v_ps);
system(pad_4displ);

if(mthd=='g')
{
  system(copy_cs);
  system(copy_ps);
  system(pad4x8_cs);
  system(pad4x8_ps);
}

/*-----------------------------------*/
/* Truthed template correlated with */
/* unmodified gray-scale scene.     */
/*-----------------------------------*/
if(mthd=='d')
{
  system(copy_cs);
  system(pad4x8_cs);
  system(r2v_old);
  system(blend_si);

  if(bin_state=='b' || bin_state=='c')
  {
    system(stat_sibland);
```

```
    /*----------------------------------------*/
    /* Retrieve the mean value for the scene. */
    /*----------------------------------------*/
    system(mean_cs);
    if((fp1 = fopen("/tmp/trac/cs_mean_data", "r"))==NULL)
    {
      printf("Error opening /tmp/trac/cs_mean_data!\n");
      exit(1);
    }
    fgets(string, 128, fp1);
    fclose(fp1);

    i = 7;
    j = 0;
    temp = string[j];
    while(temp!='\n')
    {
      term1[j] = string[i + j];
      j = j + 1;
      temp = string[i + j];
    }
    term1[j] = '\0';
    s1_mean = atof(term1);
    s1_mean = (s1_mean * factor) + s1_mean;
  }



  if(bin_state=='b')
  {
    strcpy(bin_s1, "vthresh -i /tmp/trac/curr_blend_prev ");
    strcat(bin_s1, "-o /tmp/trac/tmp_scene1 ");
    sprintf(string, "-l %f -v 1", s1_mean);
    strcat(bin_s1, string);

    strcpy(rbin_s1, "vthresh -i /tmp/trac/curr_blend_prev ");
    strcat(rbin_s1, "-o /tmp/trac/srev_bin_4stat ");
    sprintf(string, "-u %f -v 1", s1_mean);
    strcat(rbin_s1, string);

    system(bin_s1);
    system(rbin_s1);
    system(gs1_4stat);
  }

  for(n=0; n<2; n++)
  {
    length = 24;
    for(m=0; m<targ_num; m++)
    {
      /*----------------------------------------*/
      /* Retrieve the upper left corner x and y */
      /* coordinates of the bounding rectangle. */
      /*----------------------------------------*/
      for(i=0; i<3; i++)
      {
        term1[i] = tru_cntr[count - n][length + i];
      }
      term1[i] = '\0';
      x_term = atoi(term1);
      i = i + 1;

      for(j=0; j<3; j++)
```

```
  {
    term2[j] = tru_cntr[count - n][length + i + j];
  }
  term2[j] = '\0';
  y_term = atoi(term2);
  y_term = 512 - y_term;
  i = i + j + 1;

  /*-------------------------------*/
  /* Retrieve the width and height */
  /* of the bounding rectangle.    */
  /*-------------------------------*/
  for(k=0; k<2; k++)
  {
    term1[k] = tru_cntr[count -n][length + i + k + 2];
  }
  term1[k] = '\0';
  width = atoi(term1);
  k = k + 1;

  for(l=0; l<2; l++)
  {
    term2[l] = tru_cntr[count -n][length +i +k +l +4];
  }
  term2[l] = '\0';
  height = atoi(term2);
  length = length + i + k + l + 23;
}

/*-------------------------------*/
/* Extract true binarized target */
/* from current scene.           */
/*-------------------------------*/
if(x_term!=0 && y_term!=0 && targ_num!=0)
{
  for(i=1; i>n; i--)
  {
    strcpy(extru_s1, "vextract -i /tmp/trac/tmp_scene1");
    strcat(extru_s1, " -o /tmp/trac/tru_rect_s1 ");
    sprintf(string, "-x %d -y %d ", x_term, y_term);
    strcat(extru_s1, string);
    sprintf(string, "-w %d -h %d", width, height);
    strcat(extru_s1, string);

    strcpy(intru_s1, "vinsert -i1 blank_320x499 ");
    strcat(intru_s1, "-i2 /tmp/trac/tru_rect_s1 ");
    strcat(intru_s1, "-o /tmp/trac/tmp_scene1 ");
    sprintf(string, "-x %d -y %d", x_term, y_term);
    strcat(intru_s1, string);

    system(extru_s1);
    system(intru_s1);
    system(blend_x_s1);
    system(cs1_4stat);
  }

  /*-----------------------------------------*/
  /* Extract true target from previous scene. */
  /*-----------------------------------------*/
  for(i=0; i<n; i++)
  {
    strcpy(extru_t1, "vextract -i /tmp/trac/temp1_viff ");
    strcat(extru_t1, "-o /tmp/trac/tru_rect_t1 ");
    sprintf(string, "-x %d -y %d ", x_term, y_term);
    strcat(extru_t1, string);
```


(17

```
        sprintf(string, "-w %d -h %d", width, height);
        strcat(extru_t1, string);

        strcpy(intru_t1, "vinsert -i1 blank_320x490 ");
        strcat(intru_t1, "-i2 /tmp/trac/tru_rect_t1 ");
        strcat(intru_t1, "-o /tmp/trac/temp1_viff ");
        sprintf(string, "-x %d -y %d", x_term, y_term);
        strcat(intru_t1, string);

        system(extru_t1);
        system(intru_t1);
        system(ct1_4stat);
      }
    }
  }
}

if(mthd=='e' || mthd=='f' || mthd=='g')
{
  system(r2v_old);
  system(blend_s1);
  system(blend_t1);

  if(bin_state=='b' || bin_state=='c')
  {
    system(stat_s1blend);
    system(stat_t1blend);

    /*----------------------------------------*/
    /* Retrieve the mean value for the scene. */
    /*----------------------------------------*/
    system(mean_cs);
    if((fp1 = fopen("/tmp/trac/cs_mean_data", "r"))==NULL)
    {
      printf("Error opening /tmp/trac/cs_mean_data!\n");
      exit(1);
    }
    fgets(string, 128, fp1);
    fclose(fp1);

    i = 7;
    j = 0;
    temp = string[j];
    while(temp!='\n')
    {
      term1[j] = string[i + j];
      j = j + 1;
      temp = string[i + j];
    }
    term1[j] = '\0';
    s1_mean = atof(term1);
    s1_mean = (s1_mean * factor) + s1_mean;




    /*----------------------------------------*/
    /* Retrieve the mean value for the template. */
    /*----------------------------------------*/
    system(mean_ps);
    if((fp2 = fopen("/tmp/trac/ps_mean_data", "r"))==NULL)
    {
```

```
    printf("Error opening /tmp/trac/ps_mean_data!\n");
    exit(1);
  }
  fgets(string, 128, fp2);
  fclose(fp2);

  i = 7;
  j = 0;
  temp = string[j];
  while(temp!='\n')
  {
    term2[j] = string[i + j];
    j = j + 1;
    temp = string[i + j];
  }
  term2[j] = '\0';
  ti_mean = atof(term2);
  ti_mean = (ti_mean * factor) + ti_mean;
}

/*-----------------------------------------------*/
/* Binarize blended images using image average. */
/*-----------------------------------------------*/
if(mthd=='e' && bin_state=='b')
{
  strcpy(bin_s1, "vthresh -i /tmp/trac/curr_blend_prev ");
  strcat(bin_s1, "-o /tmp/trac/tmp_scene1 ");
  sprintf(string, "-l %f -v 255", s1_mean);
  strcat(bin_s1, string);

  strcpy(bin_t1, "vthresh -i /tmp/trac/prev_blend_old ");
  strcat(bin_t1, "-o /tmp/trac/tmp_temp1 ");
  sprintf(string, "-l %f -v 255", t1_mean);
  strcat(bin_t1, string);

  system(bin_s1);
  system(bin_t1);
  system(s1_4stat);
  system(t1_4stat);
}

/*-----------------------------------*/
/* Binarize blended images using the */
/* Cline binarizattion technique.    */
/*-----------------------------------*/
if(mthd=='e' && bin_state=='c')
{
  strcpy(bin_s1, "vthresh -i /tmp/trac/curr_blend_prev ");
  strcat(bin_s1, "-o /tmp/trac/bin_4and_s1 ");
  sprintf(string, "-l %f -v 255", s1_mean);
  strcat(bin_s1, string);

  strcpy(bin_t1, "vthresh -i /tmp/trac/prev_blend_old ");
  strcat(bin_t1, "-o /tmp/trac/bin_4and_t1 ");
  sprintf(string, "-l %f -v 255", t1_mean);
  strcat(bin_t1, string);

  system(bin_s1);
  system(bin_t1);
  system(dyn1_s1);
  system(dyn1_t1);
  system(and_s1);
  system(and_t1);
  system(s1_4stat);
  system(t1_4stat);
```

```
}

/*--------------------------------*/
/* Binarize blended images using */
/* line-by-line average.         */
/*--------------------------------*/
if(mthd=='e' && bin_state=='a')
{
  line_bin("/tmp/trac/curr_blend_prev", "/tmp/trac/tmp_scene1", factor, 255, 0);
  line_bin("/tmp/trac/prev_blend_old", "/tmp/trac/tmp_temp1", factor, 255, 0);

  system(s1_4stat);
  system(t1_4stat);
}

if(mthd=='f')
{
  /*-----------------------------------------------*/
  /* Binarize blended images using image average. */
  /*-----------------------------------------------*/
  if(bin_state=='b')
  {
    strcpy(bin_s1, "vthresh -i /tmp/trac/curr_blend_prev ");
    strcat(bin_s1, "-o /tmp/trac/tmp_scene1 ");
    sprintf(string, "-l %f -v 1", s1_mean);
    strcat(bin_s1, string);

    strcpy(bin_t1, "vthresh -i /tmp/trac/prev_blend_old ");
    strcat(bin_t1, "-o /tmp/trac/tmp_temp1 ");
    sprintf(string, "-l %f -v 1", t1_mean);
    strcat(bin_t1, string);

    system(bin_s1);
    system(bin_t1);
  }

  /*--------------------------------*/
  /* Binarize blended images using */
  /* line-by-line average.         */
  /*--------------------------------*/
  else if(bin_state=='a')
  {
    line_bin("/tmp/trac/curr_blend_prev", "/tmp/trac/tmp_scene1", factor, 1, 0);
    line_bin("/tmp/trac/prev_blend_old", "/tmp/trac/tmp_temp1", factor, 1, 0);
  }

  /*-----------------------------------*/
  /* Binarize blended images using the */
  /* Cline binarizattion technique.    */
  /*-----------------------------------*/
  else if(bin_state=='c')
  {
    strcpy(bin_s1, "vthresh -i /tmp/trac/curr_blend_prev ");
    strcat(bin_s1, "-o /tmp/trac/bin_4and_s1 ");
    sprintf(string, "-l %f -v 1", s1_mean);
    strcat(bin_s1, string);

    strcpy(bin_t1, "vthresh -i /tmp/trac/prev_blend_old ");
    strcat(bin_t1, "-o /tmp/trac/bin_4and_t1 ");
    sprintf(string, "-l %f -v 1", t1_mean);
    strcat(bin_t1, string);

    system(bin_s1);
    system(bin_t1);
    system(dyn1_s1);
```

```
    system(dyn1_t1);
    system(and_s1);
    system(and_t1);
  }
}

if(mthd=='g' && bin_state=='b')
{
  strcpy(bin_s1, "vthresh -i /tmp/trac/curr_blend_prev ");
  strcat(bin_s1, "-o /tmp/trac/tmp_scene1 ");
  sprintf(string, "-l %f -v 1", s1_mean);
  strcat(bin_s1, string);

  strcpy(bin_t1, "vthresh -i /tmp/trac/prev_blend_old ");
  strcat(bin_t1, "-o /tmp/trac/tmp_temp1 ");
  sprintf(string, "-l %f -v 1", t1_mean);
  strcat(bin_t1, string);

  strcpy(rbin_s1, "vthresh -i /tmp/trac/curr_blend_prev ");
  strcat(rbin_s1, "-o /tmp/trac/srev_bin_4stat ");
  sprintf(string, "-u %f -v 1", s1_mean);
  strcat(rbin_s1, string);

  strcpy(rbin_t1, "vthresh -i /tmp/trac/prev_blend_old ");
  strcat(rbin_t1, "-o /tmp/trac/trev_bin_4stat ");
  sprintf(string, "-u %f -v 1", t1_mean);
  strcat(rbin_t1, string);

  system(bin_t1);
  system(bin_s1);
  system(rbin_t1);
  system(rbin_s1);
  system(gt1_4stat);
  system(gs1_4stat);
}

if(mthd=='g')
{
  for(n=0; n<2; n++)
  {
    length = 24;
    for(m=0; m<targ_num; m++)
    {
      /*---------------------------------------*/
      /* Retrieve the upper left corner x and y */
      /* coordinates of the bounding rectangle. */
      /*---------------------------------------*/
      for(i=0; i<3; i++)
      {
        term1[i] = tru_cntr[count - n][length + i];
      }
      term1[i] = '\0';
      x_term = atoi(term1);
      i = i + 1;

      for(j=0; j<3; j++)
      {
        term2[j] = tru_cntr[count - n][length + i + j];
      }
      term2[j] = '\0';
      y_term = atoi(term2);
      y_term = 512 - y_term;
      i = i + j + 1;

      /*-------------------------------*/
```

```
  /* Retrieve the width and height */
  /* of the bounding rectangle.    */
  /*-------------------------------*/
  for(k=0; k<2; k++)
  {
    term1[k] = tru_cntr[count -n][length +i +k +2];
  }
  term1[k] = '\0';
  width = atoi(term1);
  k = k + 1;

  for(l=0; l<2; l++)
  {
    term2[l] = tru_cntr[count -n][length +i +k +l +4];
  }
  term2[l] = '\0';
  height = atoi(term2);
  length = length + i + k + l + 23;
}

/*-------------------------------*/
/* Extract true binarized target */
/* from current scene.           */
/*-------------------------------*/
if(x_term!=0 && y_term!=0 && targ_num!=0)
{
  for(i=1; i>n; i--)
  {
    strcpy(extru_s1, "vextract -i /tmp/trac/tmp_scene1"`
    strcat(extru_s1, " -o /tmp/trac/tru_rect_s1 ");
    sprintf(string, "-x %d -y %d ", x_term, y_term);
    strcat(extru_s1, string);
    sprintf(string, "-w %d -h %d", width, height);
    strcat(extru_s1, string);

    strcpy(intru_s1, "vinsert -i1 blank_320x499 ");
    strcat(intru_s1, "-i2 /tmp/trac/tru_rect_s1 ");
    strcat(intru_s1, "-o /tmp/trac/tmp_scene1 ");
    sprintf(string, "-x %d -y %d", x_term, y_term);
    strcat(intru_s1, string);

    system(extru_s1);
    system(intru_s1);
  }

  /*-------------------------------*/
  /* Extract true binarized target */
  /* from previous scene.          */
  /*-------------------------------*/
  for(i=0; i<n; i++)
  {
    strcpy(extru_t1, "vextract -i /tmp/trac/tmp_temp1 ");
    strcat(extru_t1, "-o /tmp/trac/tru_rect_t1 ");
    sprintf(string, "-x %d -y %d ", x_term, y_term);
    strcat(extru_t1, string);
    sprintf(string, "-w %d -h %d", width, height);
    strcat(extru_t1, string);

    strcpy(intru_t1, "vinsert -i1 blank_320x499 ");
    strcat(intru_t1, "-i2 /tmp/trac/tru_rect_t1 ");
    strcat(intru_t1, "-o /tmp/trac/tmp_temp1 ");
    sprintf(string, "-x %d -y %d", x_term, y_term);
    strcat(intru_t1, string);

    system(extru_t1);
```

```
            system(intru_ti);
          }
        }
      }
    }
    if(mthd=='f' || mthd=='g')
    {
      system(blend_x_ti);
      system(blend_x_si);
      system(cti_4stat);
      system(csi_4stat);
    }

    if(mthd=='e')
    {
      swap = rename("/tmp/trac/tmp_scene1", "/tmp/trac/scene1_viff");
      if(swap!=0)
      {
        printf("File renaming operation failed!\n");
      }

      swap = rename("/tmp/trac/tmp_temp1", "/tmp/trac/temp1_viff");
      if(swap!=0)
      {
        printf("File renaming operation failed!\n");
      }
    }
  }

  /*-----------------------------------*/
  /* True target extraction algorithm. */
  /*-----------------------------------*/
  if(mthd=='c')
  {
    for(n=0; n<2; n++)
    {
      length = 24;
      for(m=0; m<targ_num; m++)
      {
        /*---------------------------------------*/
        /* Retrieve the upper left corner x and y */
        /* coordinates of the bounding rectangle. */
        /*---------------------------------------*/
        for(i=0; i<3; i++)
        {
          term1[i] = tru_cntr[count - n][length + i];
        }
        term1[i] = '\0';
        x_term = atoi(term1);
        i = i + 1;

        for(j=0; j<3; j++)
        {
          term2[j] = tru_cntr[count - n][length + i + j];
        }
        term2[j] = '\0';
        y_term = atoi(term2);
        i = i + j + 1;

        /*--------------------------------*/
        /* Retrieve the width and height */
        /* of the bounding rectangle.    */
        /*--------------------------------*/
        for(k=0; k<2; k++)
        {
```

```
      term1[k] = tru_cntr[count -n][length + i + k + 2];
    }
    term1[k] = '\0';
    width = atoi(term1);
    k = k + 1;

    for(l=0; l<2; l++)
    {
      term2[l] = tru_cntr[count -n][length +i +k +l +4];
    }
    term2[l] = '\0';
    height = atoi(term2);
    length = length + i + k + l + 23;
  }

  /*----------------------------------------*/
  /* Extract true target from current scene. */
  /*----------------------------------------*/
  if(x_term!=0 && y_term!=0 && targ_num!=0)
  {
    y_term = 512 - y_term;
    for(i=1; i>n; i--)
    {
      strcpy(extru_s1, "vextract -i /tmp/trac/scene1_viff ");
      strcat(extru_s1, "-o /tmp/trac/tru_rect_s1 ");
      sprintf(string, "-x %d -y %d ", x_term, y_term);
      strcat(extru_s1, string);
      sprintf(string, "-w %d -h %d", width, height);
      strcat(extru_s1, string);

      strcpy(intru_s1, "vinsert -i1 blank_320x499 ");
      strcat(intru_s1, "-i2 /tmp/trac/tru_rect_s1 ");
      strcat(intru_s1, "-o /tmp/trac/scene1_viff ");
      sprintf(string, "-x %d -y %d", x_term, y_term);
      strcat(intru_s1, string);

      system(extru_s1);
      system(intru_s1);
      system(cs1_4stat);
    }

    /*----------------------------------------*/
    /* Extract true target from previous scene. */
    /*----------------------------------------*/
    for(i=0; i<n; i++)
    {
      strcpy(extru_t1, "vextract -i /tmp/trac/temp1_viff ");
      strcat(extru_t1, "-o /tmp/trac/tru_rect_t1 ");
      sprintf(string, "-x %d -y %d ", x_term, y_term);
      strcat(extru_t1, string);
      sprintf(string, "-w %d -h %d", width, height);
      strcat(extru_t1, string);

      strcpy(intru_t1, "vinsert -i1 blank_320x499 ");
      strcat(intru_t1, "-i2 /tmp/trac/tru_rect_t1 ");
      strcat(intru_t1, "-o /tmp/trac/temp1_viff ");
      sprintf(string, "-x %d -y %d", x_term, y_term);
      strcat(intru_t1, string);

      system(extru_t1);
      system(intru_t1);
      system(ct1_4stat);
    }
  }
}
```

```
}

/*-------------------------*/
/* Binarization algorithm. */
/*-------------------------*/
if((mthd=='a' || mthd=='b') && bin_state!='a')
{
  system(cs_stat);
  system(ps_stat);
  system(mean_cs);

  /*-------------------------------------*/
  /* Determine mean of the current_scene. */
  /*-------------------------------------*/
  if((fp4 = fopen("/tmp/trac/cs_mean_data", "r"))==NULL)
  {
    printf("Error opening /tmp/trac/cs_mean_data!\n");
    exit(1);
  }
  fgets(string, 128, fp4);
  fclose(fp4);

  i = 7;
  j = 0;
  temp = string[j];
  while(temp!='\n')
  {
    term1[j] = string[i + j];
    j = j + 1;
    temp = string[i + j];
  }
  term1[j] = '\0';
  si_mean = atof(term1);
  si_mean = (si_mean * factor) + si_mean;

  /*-----------------------------------------*/
  /* Determine the mean of the previous_scene. */
  /*-----------------------------------------*/
  system(mean_ps);
  if((fp5 = fopen("/tmp/trac/ps_mean_data", "r"))==NULL)
  {
    printf("Error opening /tmp/trac/ps_mean_data!\n");
    exit(1);
  }
  fgets(string, 128, fp5);
  fclose(fp5);

  i = 7;
  j = 0;
  temp = string[j];
  while(temp!='\n')
  {
    term2[j] = string[i + j];
    j = j + 1;
    temp = string[i + j];
  }
  term2[j] = '\0';
  ti_mean = atof(term2);
  ti_mean = (ti_mean * factor) + ti_mean;
}

if(mthd=='a' && bin_state=='b')
{
  /*--------------------------------*/
  /* Binarize using scene average. */
```

```
  /*------------------------------*/
  strcpy(bin_s1, "vthresh -i /tmp/trac/scene1_viff ");
  strcat(bin_s1, "-o /tmp/trac/tmp_scene1 ");
  sprintf(string, "-l %f -v 255", s1_mean);
  strcat(bin_s1, string);

  strcpy(bin_t1, "vthresh -i /tmp/trac/temp1_viff ");
  strcat(bin_t1, "-o /tmp/trac/tmp_temp1 ");
  sprintf(string, "-l %f -v 255", t1_mean);
  strcat(bin_t1, string);

  system(bin_s1);
  system(bin_t1);
}

/*-----------------------------------------------*/
/* Binarize using the Cline binarization technique. */
/*-----------------------------------------------*/
if(mthd=='a' && bin_state=='c')
{
  strcpy(bin_s1, "vthresh -i /tmp/trac/scene1_viff ");
  strcat(bin_s1, "-o /tmp/trac/bin_4and_s1 ");
  sprintf(string, "-l %f -v 255", s1_mean);
  strcat(bin_s1, string);

  strcpy(bin_t1, "vthresh -i /tmp/trac/temp1_viff ");
  strcat(bin_t1, "-o /tmp/trac/bin_4and_t1 ");
  sprintf(string, "-l %f -v 255", t1_mean);
  strcat(bin_t1, string);

  system(bin_s1);
  system(bin_t1);
  system(dyn2_s1);
  system(dyn2_t1);
  system(and_s1);
  system(and_t1);
}

/*--------------------------------------*/
/* Binarize using line-by-line average. */
/*--------------------------------------*/
if(mthd=='a' && bin_state=='a')
{
  line_bin("/tmp/trac/scene1_viff", "/tmp/trac/tmp_scene1", factor, 255, 0);
  line_bin("/tmp/trac/temp1_viff", "/tmp/trac/tmp_temp1", factor, 255, 0);
}

if(mthd=='a')
{
  system(t1_4stat);
  system(s1_4stat);

  swap = rename("/tmp/trac/tmp_scene1", "/tmp/trac/scene1_viff");
  if(swap!=0)
  {
    printf("File renaming operation failed!\n");
  }

  swap = rename("/tmp/trac/tmp_temp1", "/tmp/trac/temp1_viff");
  if(swap!=0)
  {
    printf("File renaming operation failed!\n");
  }
}
```

```
if(mthd=='b' && bin_state=='b')
{
  /*------------------------------*/
  /* Binarize using scene average. */
  /*------------------------------*/
  strcpy(bin_s1, "vthresh -i /tmp/trac/scene1_viff ");
  strcat(bin_s1, "-o /tmp/trac/tmp_scene1 ");
  sprintf(string, "-l %f -v 1", s1_mean);
  strcat(bin_s1, string);

  strcpy(bin_t1, "vthresh -i /tmp/trac/temp1_viff ");
  strcat(bin_t1, "-o /tmp/trac/tmp_temp1 ");
  sprintf(string, "-l %f -v 1", t1_mean);
  strcat(bin_t1, string);

  system(bin_s1);
  system(bin_t1);
  system(s1_x_bin);
  system(t1_x_bin);
  system(ct1_4stat);
  system(cs1_4stat);
}

/*-----------------------------------------------*/
/* Binarize using Cline binarization technique. */
/*-----------------------------------------------*/
if(mthd=='b' && bin_state=='c')
{
  strcpy(bin_s1, "vthresh -i /tmp/trac/scene1_viff ");
  strcat(bin_s1, "-o /tmp/trac/bin_4and_s1 ");
  sprintf(string, "-l %f -v 1", s1_mean);
  strcat(bin_s1, string);

  strcpy(bin_t1, "vthresh -i /tmp/trac/temp1_viff ");
  strcat(bin_t1, "-o /tmp/trac/bin_4and_t1 ");
  sprintf(string, "-l %f -v 1", t1_mean);
  strcat(bin_t1, string);

  system(bin_s1);
  system(bin_t1);
  system(dyn2_s1);
  system(dyn2_t1);
  system(and_s1);
  system(and_t1);
  system(s1_x_bin);
  system(t1_x_bin);
  system(ct1_4stat);
  system(cs1_4stat);
}

/*--------------------------------------*/
/* Binarize using line-by-line average. */
/*--------------------------------------*/
if(mthd=='b' && bin_state=='a')
{
  line_bin("/tmp/trac/scene1_viff", "/tmp/trac/tmp_scene1", factor, 1, 0);
  line_bin("/tmp/trac/temp1_viff", "/tmp/trac/tmp_temp1", factor, 1, 0);

  system(s1_x_bin);
  system(t1_x_bin);
  system(ct1_4stat);
  system(cs1_4stat);
}

/*-----------------------------------------------*/
```

```
/* Correlate target template with current scene. */
/*-------------------------------------------*/
if(mthd!='g' && mthd!='d')
{
  system(pad4x8_cs);
  system(pad4x8_ps);
}

if(mthd=='d')  system(pad4x8_ps);

system(extr_s1);
system(extr_t1);
system(stat_st1);
system(stat_ss1);

system(padt1);


if(run==0 && display=='y')
{
  /*---------------------------------------*/
  /* Display template and scene sub-images. */
  /*---------------------------------------*/
  if(size!='f')
  {
    system(display_s1);
  }
  system(display_t1);
  run = run + 1;
}

/*-----------------------------------*/
/* Pad sub-image with zeros for FFT. */
/*-----------------------------------*/
if(size=='l')
{
  system(pad256_s1);
  system(pad256_t1);
}
else if(size=='s' || size=='a')
{
  system(pad64_s1);
  system(pad64_t1);
}
else if(size=='m')
{
  system(pad128_s1);
  system(pad128_t1);
}
else
{
system(pad512_s1);
tmpx = 256 - (x_template1_size / 2);
tmpy = 256 - (y_template1_size / 2);
strcpy(pad512_t1, "vpad -i /tmp/trac/sub_temp1 ");
strcat(pad512_t1, "-o /tmp/trac/sub_temp1_pad -r 512 -c 512 ");
sprintf(string, "-d %d ", tmpy);
strcat(pad512_t1, string);
sprintf(string, "-e %d -j 0 -k 0", tmpx);
strcat(pad512_t1, string);
system(pad512_t1);
}

system(fft_s1);
system(fft_t1);
```

```
system(conjug_t1);

/*----------------------------------------*/
/* The following concludes the correlation */
/* for the full search scene option.       */
/*----------------------------------------*/
if(size=='f')
{
  x_template1 = x_template1 - 3;
  y_template1 = y_template1 - 3;
  x_template1_size = x_template1_size + 6;
  y_template1_size = y_template1_size + 6;

  if(edger!='y')
  {
    strcpy(extr_t1, "vextract -i /tmp/trac/temp1_pad_4x8 ");
    strcat(extr_t1, "-o /tmp/trac/sub_temp2 ");
    sprintf(string, "-x %d ", x_template1);
    strcat(extr_t1, string);
    sprintf(string, "-y %d ", y_template1);
    strcat(extr_t1, string);
    sprintf(string, "-w %d ", x_template1_size);
    strcat(extr_t1, string);
    sprintf(string, "-h %d", y_template1_size);
    strcat(extr_t1, string);
  }
  if(edger=='y')
  {
    strcpy(extr_t1, "vdrf -i /tmp/trac/temp1_pad_4x8 -o - ");
    strcat(extr_t1, " -a1 0.35 -a2 0.35 -w 9 -t1 6 -t2 8 -l 7 ");
    strcat(extr_t1, " | vextract -i - ");
    strcat(extr_t1, "-o /tmp/trac/sub_temp2 ");
    sprintf(string, "-x %d ", x_template1);
    strcat(extr_t1, string);
    sprintf(string, "-y %d ", y_template1);
    strcat(extr_t1, string);
    sprintf(string, "-w %d ", x_template1_size);
    strcat(extr_t1, string);
    sprintf(string, "-h %d", y_template1_size);
    strcat(extr_t1, string);
  }


  system(extr_t1);

  tmpx = 256 - (x_template1_size / 2);
  tmpy = 256 - (y_template1_size / 2);
  strcpy(pad512_t1, "vpad -i /tmp/trac/sub_temp2 ");
  strcat(pad512_t1, "-o /tmp/trac/sub_temp2_pad -r 512 -c 512 ");
  sprintf(string, "-d %d ", tmpy);
  strcat(pad512_t1, string);
  sprintf(string, "-e %d -j 0 -k 0", tmpx);
  strcat(pad512_t1, string);
  system(pad512_t1);

  system(fft_t2);

  system(corrtt);
  if((fp1 = fopen("/tmp/trac/tmp_3", "r"))==NULL)
  {
    printf("Error opening /tmp/trac/tmp_3!\n");
    exit(1);
  }

  fgets(string, 128, fp1);
```

```
normt = 0;
for (yy = 0; yy <= 6; ++yy) {
  for (xx = 0; xx <= 6; ++xx) {

    if((fscanf(fp1, "%f", &cortemp[xx][yy]))==NULL)
    {
      printf("Error reading from /tmp/trac/tmp_3!\n");
      exit(1);
    }
  normt = normt + (cortemp[xx][yy] * cortemp[xx][yy]);
  }
  fscanf(fp1, "%*s");
}
fclose(fp1);

system(corrts);
if((fp1 = fopen("/tmp/trac/tmp_3", "r"))==NULL)
{
  printf("Error opening /tmp/trac/tmp_3!\n");
  exit(1);
}

fgets(string, 128, fp1);

for (yy = 0; yy <= 319; ++yy) {
  for (xx = 0; xx <= 498; ++xx) {

    if((fscanf(fp1, "%f", &cordata[xx][yy]))==NULL)
    {
      printf("Error reading from /tmp/trac/tmp_3!\n");
      exit(1);
    }
  }
  fscanf(fp1, "%*s");
}
fclose(fp1);

for (xx = 0; xx <= 9; ++ xx) {
  xtar[xx] = 0;
  ytar[xx] = 0;
  magerr[xx] = 0.22;
}
pointer = 0;

tmpx = 0;
tmpy = 0;
percent = 100.00;
for (xx = 0; xx <= 492; ++xx) {
  for (yy = 0; yy <= 312; ++yy) {
    holder = 0.00;
    holder2 = 0.00;
    count = 0;
    normc = 0.00;
    xcen = xx + 3;
    ycen = yy + 3;

    if(cordata[xcen][ycen] > (cortemp[3][3] * 0.25))
    {

      for (cxx = 0; cxx <= 6; ++cxx) {
        for (cyy = 0; cyy <= 6; ++cyy) {
          txx = cxx + xx;
          tyy = cyy + yy;
          if(cordata[txx][tyy] > cordata[xcen][ycen])
```

```
    {
      count = 1;
    }
    normc = normc + (cordata[txx][tyy] * cordata[txx][tyy]);
    norm1 = cordata[txx][tyy];
    norm2 = cortemp[cxx][cyy];
    norm3 = cortemp[3][3];
    fact = 1;
    if(cxx==3) fact = fact * 8;
    if((cxx==2) || (cxx==4)) fact = fact * 4;
    if((cxx==1) || (cxx==5)) fact = fact * 2;
    if(cyy==3) fact = fact * 8;
    if((cyy==2) || (cyy==4)) fact = fact * 4;
    if((cyy==1) || (cyy==5)) fact = fact * 2;
    if((cxx==3) && (cyy==3)) fact = fact * 7;

    holder2 = holder2 + (((float)fact) * ((norm1 - norm2) * (norm1 - norm2)));
  }
}
holder = ((sqrt((double)holder2)) / ((double)norm3));

if((count==0) && (holder < percent))
{
  norm1 = ((float)(xcen - permx));
  norm2 = ((float)(ycen - permy));
  dist = sqrt((double)((norm1 * norm1) + (norm2 * norm2)));
  if(dist < 50)
  {
    tmpx = xx + 3;
    tmpy = yy + 3;
    percent = holder;
  }
}

if(count == 0)
{
  dist = 0.00;
  normalt = sqrt((double)normt);
  normalc = sqrt((double)normc);
  for (cxx = 0; cxx <= 6; ++cxx) {
    for (cyy = 0; cyy <=6; ++cyy) {
      txx = cxx +xx;
      tyy = cyy +yy;
      normal1 = ((double)cordata[txx][tyy]) / normalc;
      normal2 = ((double)cortemp[cxx][cyy]) / normalt;
      normal3 = normal1 - normal2;
      dist = dist + (normal3 * normal3);
    }
  }
  dist = sqrt(dist);

  if(dist < magerr[pointer])
  {
    xtar[pointer] = xx + 3;
    ytar[pointer] = yy + 3;
    magerr[pointer] = dist;
    pointer = 0;
    for (cxx = 1; cxx <= 9; ++cxx) {
      if(magerr[cxx] > magerr[pointer])
      {
        pointer = cxx;
      }
    }
  }
```

```
      }

    }

  }
}

norm1 = permx - tmpx;
norm2 = permy - tmpy;
norm3 = (norm1 * norm1) + (norm2 * norm2);
dist = sqrt ((double)norm3);

for (xx =0; xx <= 9; ++ xx) {
  norm1 = permx - xtar[xx];
  norm2 = permy - ytar[xx];
  norm3 = (norm1 * norm1) + (norm2 * norm2);
  normal1 = sqrt ((double)norm3);
  if((normal1 < dist) && (normal1 < 30))
  {
    tmpx = xtar[xx];
    tmpy = ytar[xx];
    dist = normal1;
    percent = 50.0;
  }
}


if(tmpx > 2 && tmpy > 2)
{
  x_coord = tmpx + 150;
  y_coord = tmpy + 40;
}
else
{
  x_coord = permx + 150;
  y_coord = permy + 40;
  percent = 100.00;
}
}

/*-------------------------------------*/
/* The following is for all but full   */
/* field of view for search scene.     */
/*-------------------------------------*/
if(size!='f')
{
  system(multi);
  system(inv_fft);

  /*-------------------------------------*/
  /* Quadrant flip the correlation plane. */
  /*-------------------------------------*/
  if(size=='l')
  {
    system(q_flip128);
  }
  else if(size=='s' || size=='a')
  {
    system(q_flip32);
  }
  else
  {
    system(q_flip64);
  }
```

```
system(c2real);
system(Cstat);

system(corr_max);
if((fp1 = fopen("/tmp/trac/max_data", "r"))==NULL)
{
  printf("Error opening /tmp/trac/max_data!\n");
  exit(0);
}
fgets(string, 128, fp1);
fclose(fp1);
l = 9;
m = 0;
temp = string[l];
while(temp!='a')
{
  Cmax[m] = temp;
  m = m + 1;
  temp = string[l + m];
}
Cmax[m] = '\0';

system(corr_min);
if((fp1 = fopen("/tmp/trac/min_data", "r"))==NULL)
{
  printf("Error opening /tmp/trac/min_data!\n");
  exit(0);
}
fgets(string, 128, fp1);
fclose(fp1);
l = 9;
m = 0;
temp = string[l];
while(temp!='a')
{
  Cmin[m] = temp;
  m = m + 1;
  temp = string[l + m];
}
Cmin[m] = '\0';

system(corr_mean);
if((fp1 = fopen("/tmp/trac/mean_data", "r"))==NULL)
{
  printf("Error opening /tmp/trac/mean_data!\n");
  exit(0);
}
fgets(string, 128, fp1);
fclose(fp1);
l = 7;
m = 0;
temp = string[l];
while(temp!='\n')
{
  Cmean[m] = temp;
  m = m + 1;
  temp = string[l + m];
}
Cmean[m] = '\0';

system(corr_var);
if((fp1 = fopen("/tmp/trac/var_data", "r"))==NULL)
{
  printf("Error opening /tmp/trac/var_data!\n");
  exit(0);
```

```
}
fgets(string, 128, fp1);
fclose(fp1);
l = 11;
m = 0;
temp = string[l];
while(temp!='\n')
{
  Cvar[m] = temp;
  m = m + 1;
  temp = string[l + m];
}
Cvar[m] = '\0';

system(corr_stdev);
if((fp1 = fopen("/tmp/trac/stdev_data", "r"))==NULL)
{
  printf("Error opening /tmp/trac/stdev_data!\n");
  exit(0);
}
fgets(string, 128, fp1);
fclose(fp1);
l = 11;
m = 0;
temp = string[l];
while(temp!='\n')
{
  Cstdev[m] = temp;
  m = m + 1;
  temp = string[l + m];
}
Cstdev[m] = '\0';

system(byte_flip);
system(stat_pk);

/*-------------------------------------------------*/
/* Read correlation information generated by KHORDS. */
/*-------------------------------------------------*/
if((fp1 = fopen("/tmp/trac/peak_coords_1", "r"))==NULL)
{
  printf("Error opening /tmp/trac/peak_coords_1!\n");
  exit(1);
}
j = k = l = 0;
for(i=0; i<12; i++)  fgets(string, 128, fp1);
fclose(fp1);

temp = string[offset + j];
while(temp!='(')
{
  j = j + 1;
  temp = string[offset + j];
}

/*-------------------------------------------------*/
/* Retrieve x- correlation peak coordinate.        */
/*-------------------------------------------------*/
j = j + 1;
temp = string[offset + j];
while(temp!=',')
{
  term1[k] = temp;
  j = j + 1;
  k = k + 1;
```

```
    temp = string[offset + j];
  }
  term1[k] = '\0';

  /*-------------------------------------------------*/
  /* Retrieve y- correlation peak coordinate.        */
  /*-------------------------------------------------*/
  j = j + 1;
  temp = string[offset + j];
  while(temp!=')')
  {
    term2[l] = temp;
    l = l + 1;
    j = j + 1;
    temp = string[offset + j];
  }
  term2[l] = '\0';

  /*-----------------------------------------*/
  /* Compute new target location coordinates. */
  /*-----------------------------------------*/
  x_peak = atoi(term1);
  y_peak = atoi(term2);
  for(m=0; m<5; m++)  term1[m] = '0';
  for(m=0; m<5; m++)  term2[m] = '0';

  /*xf_coord = ((double) x_coord);*/
  /*yf_coord = ((double) y_coord);*/

  x_coord = x_coord + (x_peak - (pad_size / 2));
  y_coord = y_coord + (y_peak - (pad_size / 2));

  f_temp = ((double) pad_size / 2.0);
  xf_coord = xf_coord + ((double) x_peak - f_temp);
  yf_coord = yf_coord + ((double) y_peak - f_temp);
}

/*-------------------------------------------------*/
/* Generate adaptive template dimensions and       */
/* location for correlation.                       */
/*-------------------------------------------------*/
  if(size=='a' || size=='f')
  {
    xedge = x_coord - 200;
    yedge = y_coord - 90;
    if(xedge < 0)
    {
      xedge = 0;
    }
    if(xedge > 398)
    {
      xedge = 398;
    }
    if(yedge < 0)
    {
      yedge = 0;
    }
    if(yedge > 220)
    {
      yedge = 220;
    }
    strcpy(extr_win, "vextract -i /tmp/trac/scene1_viff -o - ");
    sprintf(string, "-x %d -y %d -w 100 -h 100", xedge, yedge);
    strcat(extr_win, string);
    strcat(extr_win, " | viff2mat -i - ");
```

```
strcat(extr_win, "-o /tmp/trac/tmp_1 -j 1 -b 0 ");
strcat(extr_win, "-d 0 -f 0 ");

system(extr_win);

if((fp1 = fopen("/tmp/trac/tmp_1", "r"))==NULL)
{
  printf("Error opening /tmp/trac/tmp_1!\n");
  exit(1);
}

fgets(string, 128, fp1);

for (yy = 0; yy <= 99; ++yy) {
  for (xx = 0; xx <= 99; ++xx) {

    if((fscanf(fp1, "%d", &xy[xx][yy]))==NULL)
    {
      printf("Error reading from /tmp/trac/tmp_1!\n");
      exit(1);
    }
  }
  fscanf(fp1, "%*s");
}
fclose(fp1);

tmpx = x_coord - 150 - xedge;
tmpy = y_coord - 40 - yedge;

/*-------------------------------------------------*/
/* Check to see if current selected pixel is part */
/* of a target.                                    */
/*-------------------------------------------------*/

if(xy[tmpx][tmpy] <= 70)
{
  count = 0;
  a1 = tmpy;
  b1 = tmpx;
  c1 = tmpy;
  d1 = tmpx;
  while(count == 0)
  {
    a1 = a1 - 1;
    b1 = b1 + 1;
    c1 = c1 + 1;
    d1 = d1 - 1;
    if(a1 < 0) a1 = 0;
    if(b1 > 100) b1 = 100;
    if(c1 > 100) c1 = 100;
    if(d1 < 0) d1 = 0;


    for(xx = d1; xx <= b1; ++xx) {
      if(xy[xx][a1] >= 71)
      {
        count = 1;
        if(xy[xx][a1] >= (xy[tmpx][tmpy] + 1))
        {
          tmpx = xx;
          tmpy = a1;
        }
      }
    }
```

```
    for(yy = a1; yy <= c1; ++yy) {
      if(xy[b1][yy] >= 71)
      {
        count = 1;
        if(xy[b1][yy] >= (xy[tmpx][tmpy] + 1))
        {
          tmpx = b1;
          tmpy = yy;
        }
      }
    }

    for(xx = d1; xx <= b1; ++xx) {
      if(xy[xx][c1] >= 71)
      {
        count = 1;
        if(xy[xx][c1] >= (xy[tmpx][tmpy] + 1))
        {
          tmpx = xx;
          tmpy = c1;
        }
      }
    }

    for(yy = a1; yy <= c1; ++yy) {
      if(xy[d1][yy] >= 71)
      {
        count = 1;
        if(xy[d1][yy] >= (xy[tmpx][tmpy] + 1))
        {
          tmpx = d1;
          tmpy = yy;
        }
      }
    }
    if((a1==0)&&(b1==99)&&(c1==99)&&(d1==0))
    {
      tmpx = permx;
      tmpy = permy;
    }

  }
}

/*-------------------------------------*/
/* Find bounding box of target.        */
/*-------------------------------------*/

a1 = tmpy - 1;
a2 = 0;
b1 = tmpx + 1;
b2 = 0;
c1 = tmpy + 1;
c2 = 0;
d1 = tmpx - 1;
d2 = 0;

while(a2==0 || b2==0 || c2==0 || d2==0)
{
  if(a2 == 0)
  {
    a1 = a1 - 1;
  }
  a2 = 0;
  count = 0;
```

```
if(a1 < 0)
{
  a1 = 0;
  a2 = 1;
}
for(xx = d1; xx <= b1; ++xx) {
  if(xy[xx][a1] <= 70)
  {
    count = count + 1;
  }
}
check = (count / (b1 - d1));
if(check >= 0.9)
{
  a2 = 1;
}

if(b2 == 0)
{
  b1 = b1 + 1;
}
b2 = 0;
count = 0;
if(b1 > 100)
{
  b1 = 100;
  b2 = 1;
}
for(yy = a1; yy <= c1; ++yy) {
  if(xy[b1][yy] <= 70)
  {
    count = count + 1;
  }
}
check = (count / (c1 - a1));
if(check >= 0.9)
{
  b2 = 1;
}

if(c2 == 0)
{
  c1 = c1 + 1;
}
c2 = 0;
count = 0;
if(c1 > 100)
{
  c1 = 100;
  c2 = 1;
}
for(xx = d1; xx <= b1; ++xx) {
  if(xy[xx][c1] <= 70)
  {
    count = count + 1;
  }
}
check = (count / (b1 - d1));
if(check >= 0.9)
{
  c2 = 1;
}

if(d2 == 0)
{
```

```
      d1 = d1 - 1;
    }
    d2 = 0;
    count = 0;
    if(d1 < 0)
    {
      d1 = 0;
      d2 = 1;
    }
    for(yy = a1; yy <= c1; ++yy) {
      if(xy[d1][yy] <= 70)
      {
        count = count + 1;
      }
    }
    check = (count / (c1 - a1));
    if(check >= 0.9)
    {
      d2 = 1;
    }

  }

  x_coord = ((b1 + d1) / 2) + xedge;
  y_coord = ((c1 + a1) / 2) + yedge;
  permx = x_coord;
  permy = y_coord;

  x_template1_size = b1 - d1;
  y_template1_size = c1 - a1;
  if(x_template1_size > 64)
  {
    x_template1_size = 64;
  }
  if(y_template1_size > 64)
  {
    y_template1_size = 64;
  }

  x_value = x_coord;
  y_value = y_coord;
  xf_coord = ((double) (b1 + d1) / 2) + xedge;
  yf_coord = ((double) (c1 + a1) / 2) + yedge;
  xf_cntr = xf_coord;
  yf_cntr = yf_coord;

  x_coord = 150 + x_coord;
  y_coord = 40 + y_coord;
  xf_coord = 150.0 + xf_coord;
  yf_coord = 40.0 + yf_coord;

}


/*-------------------------------------------------*/
/* Compute upper left x- and y- pixel coordinates  */
/* for subimages template1, scene1, and sub-display. */
/*-------------------------------------------------*/
x_template1 = x_coord - (x_template1_size / 2);
y_template1 = y_coord - (y_template1_size / 2);
x_scene1 = x_coord - (x_scene1_size / 2);
y_scene1 = y_coord - (y_scene1_size / 2);
if(size=='f')
{
  x_scene1 = 150;
```

```
  y_scene1 = 40;
}
x_display = 150;
y_display = 40;
x_display_size = 499;
y_display_size = 320;

if(edger!='y')
{
  strcpy(extr_s1, "vextract -i /tmp/trac/scene1_pad_4x8 ");
  strcat(extr_s1, "-o /tmp/trac/sub_scene1 ");
  sprintf(string, "-x %d -y %d ", x_scene1, y_scene1);
  strcat(extr_s1, string);
  sprintf(string, "-w %d ", x_scene1_size);
  strcat(extr_s1, string);
  sprintf(string, "-h %d", y_scene1_size);
  strcat(extr_s1, string);

  strcpy(extr_t1, "vextract -i /tmp/trac/temp1_pad_4x8 ");
  strcat(extr_t1, "-o /tmp/trac/sub_temp1 ");
  sprintf(string, "-x %d ", x_template1);
  strcat(extr_t1, string);
  sprintf(string, "-y %d ", y_template1);
  strcat(extr_t1, string);
  sprintf(string, "-w %d ", x_template1_size);
  strcat(extr_t1, string);
  sprintf(string, "-h %d", y_template1_size);
  strcat(extr_t1, string);

}

if(edger=='y')
{
  strcpy(extr_s1, "vdrf -i /tmp/trac/scene1_pad_4x8 -o - ");
  strcat(extr_s1, "-a1 0.35 -a2 0.35 -w 9 -t1 6 -t2 8 -l 7 ");
  strcat(extr_s1, " | vextract -i - ");
  strcat(extr_s1, "-o /tmp/trac/sub_scene1 ");
  sprintf(string, "-x %d -y %d ", x_scene1, y_scene1);
  strcat(extr_s1, string);
  sprintf(string, "-w %d ", x_scene1_size);
  strcat(extr_s1, string);
  sprintf(string, "-h %d", y_scene1_size);
  strcat(extr_s1, string);

  strcpy(extr_t1, "vdrf -i /tmp/trac/temp1_pad_4x8 -o - ");
  strcat(extr_t1, "-a1 0.35 -a2 0.35 -w 9 -t1 6 -t2 8 -l 7 ");
  strcat(extr_t1, " | vextract -i - ");
  strcat(extr_t1, "-o /tmp/trac/sub_temp1 ");
  sprintf(string, "-x %d ", x_template1);
  strcat(extr_t1, string);
  sprintf(string, "-y %d ", y_template1);
  strcat(extr_t1, string);
  sprintf(string, "-w %d ", x_template1_size);
  strcat(extr_t1, string);
  sprintf(string, "-h %d", y_template1_size);
  strcat(extr_t1, string);

}

strcpy(extr_sub, "vextract -i /tmp/trac/pad_4x8_displ ");
strcat(extr_sub, "-o /tmp/trac/sub_displ_data ");
sprintf(string, "-x %d ", x_display);
strcat(extr_sub, string);
sprintf(string, "-y %d ", y_display);
strcat(extr_sub, string);
```

```
sprintf(string, "-w %d ", x_display_size);
strcat(extr_sub, string);
sprintf(string, "-h %d", y_display_size);
strcat(extr_sub, string);

system(extr_sub);

  system(extr_cross);

  count = 0;
  xpnt = 0;
  ypnt = 0;
  xlen = 50;
  ylen = 50;
  tmpx = x_coord - 150;
  tmpy = y_coord - 40;
  if(tmpx < 25)
  {
    xpnt = 25 - tmpx;
    xlen = 50 - xpnt;
    count = 1;
  }
  if(tmpx > 474)
  {
    xlen = 50 - 499 + tmpx;
    count = 1;
  }
  if(tmpy < 25)
  {
    ypnt = 25 - tmpy;
    ylen = 50 - ypnt;
    count = 1;
  }
  if(tmpy > 295)
  {
    ylen = 50 - 320 + tmpy;
    count = 1;
  }
  if(count==1)
  {
    strcpy(cut_cross, "vextract -i /tmp/trac/cross_data ");
    strcat(cut_cross, "-o /tmp/trac/cross_dat ");
    sprintf(string, "-x %d ", xpnt);
    strcat(cut_cross, string);
    sprintf(string, "-y %d ", ypnt);
    strcat(cut_cross, string);
    sprintf(string, "-w %d ", xlen);
    strcat(cut_cross, string);
    sprintf(string, "-h %d ", ylen);
    strcat(cut_cross, string);

    system(cut_cross);
  }

  xedge = tmpx - 25 + xpnt;
  yedge = tmpy - 25 + ypnt;
  if(count == 1)
  {
    sprintf(string, "vpad -i /tmp/trac/cross_dat ");
    strcpy(pad_cross, string);
    strcat(pad_cross, "-o /tmp/trac/crosshair -r 320 -c 499 ");
    sprintf(string, "-d %d ", yedge);
    strcat(pad_cross, string);
    sprintf(string, "-e %d -j 0 -k 0 ", xedge);
    strcat(pad_cross, string);
```

```
  }
  else
  {
    sprintf(string, "vpad -i /tmp/trac/cross_data ");
    strcpy(pad_cross, string);
    strcat(pad_cross, "-o /tmp/trac/crosshair -r 320 -c 490 ");
    sprintf(string, "-d %d ", ydge);
    strcat(pad_cross, string);
    sprintf(string, "-e %d -j 0 -k 0 ", xedge);
    strcat(pad_cross, string);
  }
  system(pad_cross);

  strcpy(add_cross, "vadd -i1 /tmp/trac/sub_displ_data ");
  strcat(add_cross, "-i2 /tmp/trac/crosshair ");
  strcat(add_cross, "-o /tmp/trac/sub_displ_conv");

system(add_cross);
system(byte_sub);

/*-------------------------------------------------*/
/* Compute and display statistical information above */
/* the on-going adaptive correlation process.       */
/*-------------------------------------------------*/
for(m=0; m<6; m++)  term2[m] = tru_cntr[count][m];
term2[m] = '\0';
tr_info = atoi(term2);

if(targ_num!=0)
{
  for(i=0; i<targ_num; i++)
  {
    /*-----------------------------*/
    /* Retrieve the x- coordinate */
    /* of target locations.       */
    /*-----------------------------*/
    j = 0;
    temp = tru_cntr[count][m];
    while(temp!=',')
    {
      term1[j] = temp;
      j = j + 1;
      temp = tru_cntr[count][m + j];
    }
    term1[j] = '\0';
    x_numb = atoi(term1);
    xf_numb = ((double) x_numb - 0.0);
    j = j + 1;

    /*-----------------------------*/
    /* Retrieve the y- coordinate */
    /* of target locations.       */
    /*-----------------------------*/
    for(k=0; k<7; k++)
    {
      term2[k] = tru_cntr[count][m + j + k];
    }
    term2[k] = '\0';
    y_numb = atoi(term2);
    yf_numb = ((double) y_numb - 0.0);
    m = m + j + k + 21;
  }

  system(cs_pad4x8);
  system(ps_pad4x8);
```

```
  if(index==0)
  {
    printf("\n");
    system(clr_scr);

    printf("--------------------");
    printf("--- CORRELATION STATISTICS:  ");
    printf("Run_number = %s , ", run_num);
    printf("Target_Number = %d ---", targ_num);
    printf("--------------------\n");

    printf("Tracking Method:      %-37.37s", str0);
    printf("Init Window Size:  %-32.32s\n", str1);
    printf("Threshold Technique:  %-37.37s", str2);
    printf("Threshold factor:  %-.2f ", factor);
    printf("above computed threshold\n\n");

    printf("Targ     Correlate      ");
    printf(" True       Track  Err Path");
    printf("  Correlation\n");

    printf("Range    Targ_Cntr     Targ_Cntr");
    printf("    Error  (Degree)   Error\n");


    printf("---------------------------------");
    printf("---------------------------------");
    printf("---------------------------------");
    printf("-------\n");

    index = 15;
  }

  swap = rename("/tmp/trac/ready_4displ", "/tmp/trac/FOV_displ");
  if(swap!=0)
  {
    printf("File renaming operation #1 failed!\n");
  }
}
else if(targ_num==0)
{
  swap = rename("/tmp/trac/ready_4displ", "/tmp/trac/FOV_displ");
  if(swap!=0)
  {
    printf("File renaming operation #2 failed!\n");
  }

  if(index==0)
  {
    printf("\n");
    system(clr_scr);

    printf("--------------------\n");
    printf("Targ      Correlate\n");
    printf("Range     Targ_Cntr\n");
    printf("--------------------\n");
    index = 19;
  }
}

/*----------------------------------------*/
/* Display correlated target coordinates. */
/*----------------------------------------*/
```

```
if(targ_num==0 || (x_term==0 && y_term==0))
{
  printf("%5d%9.1f,%5.1f", tr_info, xf_peak, yf_peak);
  printf("\n");
  index = index - 1;
}
else
{
  /*-----------------------------------------------*/
  /* Flip y_term to adjust for different coordinate */
  /* system and reduce new target coordinates from  */
  /* a reference of 400x800 to 320x499.             */
  /*-----------------------------------------------*/
  y_numb = 512 - y_numb;
  x_peak = x_coord - 150;
  y_peak = y_coord - 40;

  yf_numb = 512.0 - yf_numb;
  xf_peak = xf_coord - 150.0;
  yf_peak = yf_coord - 40.0;

  /*---------------------------------------*/
  /* Calculate magnitude of tracking error. */
  /*---------------------------------------*/
  fvalue1 = ((xf_peak - xf_numb) * (xf_peak - xf_numb));
  fvalue2 = ((yf_peak - yf_numb) * (yf_peak - yf_numb));
  result = sqrt(fvalue1 + fvalue2);
  error = result;

  if(size!='f')
  {
    percent = 0.0;
  }

  /*-------------------------------------------*/
  /* Determine the direction of the track error. */
  /*-------------------------------------------*/
  fvalue1 = xf_peak - xf_cntr;
  fvalue2 = yf_peak - yf_cntr;


  if((xf_numb - xf_peak)>0.0 && (yf_numb - yf_peak)<0.0)
  {
    fvalue1 = xf_numb - xf_peak;
    fvalue2 = yf_peak - yf_numb;
    result = atan2(fvalue2, fvalue1);
    angle = -180.0 + 180.0 * (result / pie);
  }
  else if((xf_numb - xf_peak)>0.0 && (yf_numb - yf_peak)>0.0)
  {
    fvalue1 = xf_numb - xf_peak;
    fvalue2 = yf_numb - yf_peak;
    result = atan2(fvalue2, fvalue1);
    angle = 180.0 - 180.0 * (result / pie);
  }
  else if((xf_numb - xf_peak)<0.0 && (yf_numb - yf_peak)<0.0)
  {
    fvalue1 = xf_peak - xf_numb;
    fvalue2 = yf_peak - yf_numb;
    result = atan2(fvalue2, fvalue1);
    angle = -180.0 * (result / pie);
  }
  else if((xf_numb - xf_peak)<0.0 && (yf_numb - yf_peak)>0.0)
  {
    fvalue1 = xf_peak - xf_numb;
```

```
  fvalue2 = yf_numb - yf_peak;
  result = atan2(fvalue2, fvalue1);
  angle = 180.0 * (result / pie);
}
else if((xf_numb - xf_peak)>0.0)  angle = 180.0;
else if((xf_numb - xf_peak)<0.0)  angle = 0.0;
else if((yf_numb - yf_peak)>0.0)  angle = 90.0;
else if((yf_numb - yf_peak)<0.0)  angle = -90.0;
else angle = 999.9;

/*-----------------------------------------*/
/* Display correlation statistics to screen. */
/*-----------------------------------------*/
printf("%5d%8.1f,%5.1f", tr_info, xf_peak, yf_peak);
printf("%8.1f,%5.1f%8.2f", xf_numb, yf_numb, error);
printf("%9.1f%9.3f\n", angle, percent);

index = index - 1;

/*-----------------------------------------*/
/* Record correlation summary to a file. */
/*-----------------------------------------*/
if(times==0)
{
  fprintf(fp3,"----------------------------------------");
  fprintf(fp3,"---------- CORRELATION STATISTICS SUMMAR");
  fprintf(fp3,"Y ----------------------------------");
  fprintf(fp3,"-----------\n\n");

  fprintf(fp3,"Tracking Method:          %-40.40s", str0);
  fprintf(fp3,"Window Size:      %-40.40s\n", str1);
  fprintf(fp3,"Binarization Technique:   %-40.40s", str2);
  fprintf(fp3,"Run Number:       %-3.3s\n", run_num);
  fprintf(fp3,"Threshold Factor:         %-40.2f", factor);
  fprintf(fp3,"Target Number:    %d\n\n", targ_num);

  fprintf(fp3,"Data Format:\n\n");

  fprintf(fp3, "RANGE, X-PEAK CORR, Y-PEAK CORR, ");
  fprintf(fp3, "X-TRUE, Y-TRUE, TRACK ERROR,");
  fprintf(fp3, "ERROR PATH (DEGREES), CORRELATION ERROR\n\n");
  fprintf(fp3,"----------------------------------------");
  fprintf(fp3,"----------------------------------------");
  fprintf(fp3,"-----------------------------------\n\n");


  times = times + 1;
}

/*-----------------------------------*/
/* Write computed statistics to file. */
/*-----------------------------------*/
fprintf(fp3, "%5d%8.1f%8.1f", tr_info, xf_peak, yf_peak);
fprintf(fp3, "%8.1f%8.1f", xf_numb, yf_numb);
fprintf(fp3, "%8.2f%9.1f%9.3f\n", error, angle , percent);

fprintf(fp6, "%5d%8.2f\n", tr_info, error);

fprintf(fp7, "%5d%9d%9d\n", tr_info, x_peak, y_peak);

fprintf(fp8, "%d ", tr_info);
for (xx =0; xx <= 9; ++ xx) {
  dist = 20.0;
  for(yy = 0; yy <= 9; ++yy) {
    if((holdx[yy] != 0) && (holdy[yy] != 0))
```

```
      {
        norm1 = ((float)(xtar[xx] - holdx[yy]));
        norm2 = ((float)(ytar[xx] - holdy[yy]));
        holder = sqrt((double)((norm1 * norm1) + (norm2 * norm2)));
        if(holder < dist) dist = holder;
      }
    }
    if((xtar[xx] != 0) && (ytar[xx] != 0) && (dist <= 15.0)) {
      fprintf(fp8, " %d,%d ", xtar[xx], ytar[xx]);
    }
  }
  fprintf(fp8, "\n");

  fflush(fp3);
  fflush(fp6);
  fflush(fp7);
  fflush(fp8);

  for (xx =0; xx <= 9; ++ xx) {
    holdx[xx] = xtar[xx];
    holdy[xx] = ytar[xx];
  }

}

count = count + 1;

swap = rename("/tmp/trac/previous_scene", "/tmp/trac/old_scene");
if(swap!=0)
{
  printf("File renaming operation #3 failed!\n");
  exit(1);
}
swap=rename("/tmp/trac/current_scene", "/tmp/trac/previous_scene");
if(swap!=0)
{
  printf("File renaming operation #4 failed!\n");
  exit(1);
}

strcpy(displ_file, plotname);
sprintf(term1, "T1_%d%d%d", digit1, digit2, digit3);
strcat(displ_file, term1);
strcpy(saveimg, "cp /tmp/trac/sub_temp1 ");
sprintf(term3, "%s", displ_file);
strcat(saveimg, term3);
system(saveimg);

if(size!='f')
{
  strcpy(displ_file, plotname);
  sprintf(term1, "S1_%d%d%d", digit1, digit2, digit3);
  strcat(displ_file, term1);
  strcpy(saveimg, "cp /tmp/trac/sub_scene1");
  sprintf(term3, " %s", displ_file);
  strcat(saveimg, term3);
  system(saveimg);
}



digit3 = digit3 + 1;
if(digit3==10)
{
  digit3 = 0;
```

```
  digit2 = digit2 + 1;
}
if(digit2==10)
{
  digit2 = 0;
  digit1 = 1;
}

/*----------------------------------------*/
/* Set flag to trigger the display of the */
/* correlator's options menu.             */
/*----------------------------------------*/
if(cycle==0)
{
  if((fp1 = fopen("/tmp/trac/start2_flag", "w"))==NULL)
  {
    printf("Error opening /tmp/trac/start2_flag!\n");
    exit(1);
  }
  fputc(on, fp1);
  fclose(fp1);
  cycle = cycle + 1;
}

if((fp2 = fopen("/tmp/trac/option_flag", "r"))==NULL)
{
  printf("Error opening /tmp/trac/option_flag!\n");
  exit(1);
}
opt_flag = fgetc(fp2);
fclose(fp2);

/*------------------------------------------------*/
/* Reset opt_flag and determine the option selected. */
/*------------------------------------------------*/
if(opt_flag==on)
{
  if((fp4 = fopen("/tmp/trac/option_flag", "w"))==NULL)
  {
    printf("Error opening /tmp/trac/option_flag!\n");
    exit(1);
  }
  fputc(off, fp4);
  fclose(fp4);

  if((fp5 = fopen("/tmp/trac/option_info", "r"))==NULL)
  {
    printf("Error opening /tmp/trac/option_info!\n");
    exit(1);
  }
  opt = fgetc(fp5);

  /*----------------------------------------*/
  /* Annotate change in binarization factor. */
  /*----------------------------------------*/
  if(opt=='t')
  {
    fgets(string, 128, fp5);
    fgets(string, 128, fp5);
    factor = atof(string);
    if(targ_num!=0)
    {
      fprintf(fp3,"********************* Chan");
      fprintf(fp3,"ges in binarization threshold");
      fprintf(fp3,". New threshold factor: ");
```

```
      fprintf(fp3," %d **********\n\n", factor);
    }
  }
  fclose(fp5);
}

/*-----------------------------------------------*/
/* Annotate change in template and scene sizes. */
/*-----------------------------------------------*/
if(opt=='l' || opt=='m' || opt=='s')
{
  if(opt=='l')
  {
    size = 'l';
    y_template_size = 40;
    x_template_size = 64;
    y_scene_size = 120;
    x_scene_size = 192;
    pad_size = 256;
    if(targ_num!=0)
    {
      fprintf(fp3,"********** Chang");
      fprintf(fp3,"e in window size. New window ");
      fprintf(fp3,"size is: Large (Template=64x");
      fprintf(fp3,"40, Scene=192x120) **********");
      fprintf(fp3,"**********\n\n");
    }
  }

  if(opt=='s')
  {
    size = 's';
    y_template_size = 20;
    x_template_size = 30;
    y_scene_size = 38;
    x_scene_size = 60;
    pad_size = 64;
    if(targ_num!=0)
    {
      fprintf(fp3,"********** Chang");
      fprintf(fp3,"e in window size. New window ");
      fprintf(fp3,"size is: Small (Template=30x");
      fprintf(fp3,"20, Scene=60x38) **********");
      fprintf(fp3,"**********\n\n");
    }
  }

  if(opt=='m')
  {
    size = 'm';
    y_template_size = 28;
    x_template_size = 42;
    y_scene_size = 72;
    x_scene_size = 116;
    pad_size = 128;
    if(targ_num!=0)
    {
      fprintf(fp3,"********** Chang");
      fprintf(fp3,"e in window size. New window ");
      fprintf(fp3,"size is: Medium (Template=42x");
      fprintf(fp3,"28, Scene=116x72) **********");
      fprintf(fp3,"**********\n\n");
    }
  }
```

```
if(opt=='a')
{
  size = 'a';
  y_template1_size = 20;
  x_template1_size = 30;
  y_scene1_size = 38;
  x_scene1_size = 60;
  pad_size = 64;
  if(targ_num!=0)
  {
    fprintf(fp3,"********************** Chang");
    fprintf(fp3,"e in window size. New window ");
    fprintf(fp3,"size is: Adapt/Small (Template=");
    fprintf(fp3,"Adapt, Scene=60x38) ***********");
    fprintf(fp3,"*****************\n\n");
  }
}

if(opt=='f')
{
  size = 'f';
  y_template1_size = 20;
  x_template1_size = 30;
  y_scene1_size = 320;
  x_scene1_size = 499;
  pad_size = 512;
  if(targ_num!=0)
  {
    fprintf(fp3,"********************** Chang");
    fprintf(fp3,"e in window size. New window ");
    fprintf(fp3,"size is: Adapt/Full (Template=");
    fprintf(fp3,"Adapt, Scene=499x330) ***********");
    fprintf(fp3,"*****************\n\n");
  }
}



/*-------------------------------------*/
/* Reinitialize the KHOROS commands to */
/* extract the template and scene.     */
/*-------------------------------------*/
x_template1 = x_coord - (x_template1_size / 2);
y_template1 = y_coord - (y_template1_size / 2);
x_scene1 = x_coord - (x_scene1_size / 2);
y_scene1 = y_coord - (y_scene1_size / 2);

strcpy(extr_s1, "vextract -i /tmp/trac/scene1_pad_4x8 ");
strcat(extr_s1, "-o /tmp/trac/sub_scene1 ");
sprintf(string, "-x %d ", x_scene1);
strcat(extr_s1, string);
sprintf(string, "-y %d ", y_scene1);
strcat(extr_s1, string);
sprintf(string, "-w %d ", x_scene1_size);
strcat(extr_s1, string);
sprintf(string, "-h %d", y_scene1_size);
strcat(extr_s1, string);

strcpy(extr_t1, "vextract -i /tmp/trac/temp1_pad_4x8 ");
strcat(extr_t1, "-o /tmp/trac/sub_temp1 ");
sprintf(string, "-x %d ", x_template1);
strcat(extr_t1, string);
sprintf(string, "-y %d ", y_template1);
strcat(extr_t1, string);
sprintf(string, "-w %d ", x_template1_size);
```

```
  strcat(extr_ti, string);
  sprintf(string, "-h %d", y_template_size);
  strcat(extr_ti, string);
}

/*-------------------------------------------*/
/* Annotate change in binarization technique. */
/*-------------------------------------------*/
if(opt=='z')
{
  bin_state = 'a';
  opt = 't';
  if(targ_num!=0)
  {
    fprintf(fp3,"****************************** Change in");
    fprintf(fp3," binarization method. The new method is:");
    fprintf(fp3," Line-by-line Average. ****************");
    fprintf(fp3,"*************\n\n");
  }
}

if(opt=='b')
{
  bin_state = 'b';
  opt = 't';
  if(targ_num!=0)
  {
    fprintf(fp3,"********************************* Change");
    fprintf(fp3," in binarization method. The new method ");
    fprintf(fp3,"is: Scene Average. ***********************");
    fprintf(fp3,"*************\n\n");
  }
}

if(opt=='c')
{
  bin_state = 'c';
  opt = 't';
  if(targ_num!=0)
  {
    fprintf(fp3,"****************************** Change");
    fprintf(fp3," in binarization method. The new method ");
    fprintf(fp3,"is: Cline Binarization. ***************");
    fprintf(fp3,"*************\n\n");
  }
}

/*-----------------------------------------------------*/
/* If last image in series has been displayed or per */
/* user's request, kill background jobs and exit.    */
/*-----------------------------------------------------*/
if(opt=='e' || (argc-count)==1)
{
  strcpy(search, "ps -aux | grep putimage | grep -v");
  strcat(search, " 'grep' > /tmp/trac/job_numbers");
  system(search);

  if((fp1 = fopen("/tmp/trac/job_numbers", "r"))==NULL)
  {
    printf("Error opening /tmp/trac/job_numbers!\n");
    exit(1);
  }
  for(i=1; i<4; i++)
  {
    fgets(string, 128, fp1);
```

```
    for(m=0; m<6; m++)   ps_value[m] = string[m + 9];
    ps_value[m] = '\0';
    if(i==1)  job1 = atoi(ps_value);
    if(i==2)  job2 = atoi(ps_value);
    if(i==3)  job3 = atoi(ps_value);
  }
  fclose(fp1);
  sprintf(k_displ_fov, "kill %d", job1);
  sprintf(k_displ_si, "kill %d", job2);
  sprintf(k_displ_ti, "kill %d", job3);
  system(k_displ_fov);
  system(k_displ_si);
  system(k_displ_ti);
  if((fp2 = fopen("/tmp/trac/EOS_flag", "w"))==NULL)
  {
    printf("Error opening /tmp/trac/EOS_flag!\n");
    exit(1);
  }
  fputc(cm, fp2);
  fclose(fp2);

  if(targ_num!=0)
  {
    fclose(fp3);
    fclose(fp6);
    fclose(fp7);

  }

  printf("%c\n", '\007');
  printf("/*-------------------------------------");
  printf("---------*/\n");
  printf("/* This program has been terminated either");
  printf(" in re- */\n");
  printf("/* ponse to the user's request or because ");
  printf("the di- */\n");
  printf("/* gitized images for this sequence have b");
  printf("een ex- */\n");
  printf("/* hausted. If the program 'trac_menu' has");
  printf(" not    */\n");
  printf("/* already been terminated, please reponse");
  printf(" with   */\n");
  printf("/* option 'e' to terminate it now.        ");
  printf("        */\n");
  printf("/*-------------------------------------");
  printf("---------*/\n");
  exit(0);
}

/*-----------------------------------------------*/
/* Reset variables and kill background processes */
/* to enable the user to select a new target.    */
/*-----------------------------------------------*/
if(opt=='n')
{
  cycle = 0;
  state = 'r';
  pass = 1;
  times = 0;
  index = 0;
  run = 0;
  opt = 'a';
  factor = 0.15;
  bin_state = 'b';
```

```
        if(targ_num!=0)
        {
          fclose(fp3);
          fclose(fp6);
          fclose(fp7);

        }

        strcpy(search, "ps -aux | grep /tmp/trac/T1_di | grep ");
        strcat(search, "putimage > /tmp/trac/job_numbers");
        system(search);
        if((fp4 = fopen("/tmp/trac/job_numbers", "r"))==NULL)
        {
          printf("Error opening /tmp/trac/job_numbers!\n");
          exit(1);
        }
        fgets(string, 128, fp4);
        for(m=0; m<6; m++)   ps_value[m] = string[m + 9];
        ps_value[m] = '\0';
        job1 = atoi(ps_value);
        fclose(fp4);

        strcpy(search, "ps -aux | grep /tmp/trac/S1_dis | grep ");
        strcat(search, "putimage > /tmp/trac/job_numbers");
        system(search);
        if((fp5 = fopen("/tmp/trac/job_numbers", "r"))==NULL)
        {
          printf("Error opening /tmp/trac/job_numbers!\n");
          exit(1);
        }
        fgets(string, 128, fp5);
        for(m=0; m<6; m++)   ps_value[m] = string[m + 9];
        ps_value[m] = '\0';
        job2 = atoi(ps_value);
        fclose(fp5);

        sprintf(k_displ_t1, "kill %d", job1);
        sprintf(k_displ_s1, "kill %d", job2);
        system(k_displ_t1);
        system(k_displ_s1);
      }
    }
  }
}

/*-----------------------------------------------*/
/* Function:  line_bin   This function performs */
/* line-by-line binarization of an image.       */
/*-----------------------------------------------*/
void line_bin(infile, outfile, factor, upper, lower)
char infile[128], outfile[128];
int upper, lower;
float factor;
{
  FILE *fp1, *fp2;
  int total = 0, l, m, max = 0;
  int num_array[8], value, r, k, number;
  char temp, term2[25], string[128], to_ascii[128];
  char to_viff[128], to_byte[128];
  float average;

  /*-----------------------------------------------*/
  /* Initialize KHOROS sub-routine command strings. */
  /*-----------------------------------------------*/
  sprintf(string, "vprdata -i %s ", infile);
```

```
strcpy(to_ascii, string);
strcat(to_ascii, "-f /tmp/trac/line_bin_indata -m 0");

strcpy(to_viff, "asc2viff -i /tmp/trac/line_bin_outdata -o ");
strcat(to_viff, "/tmp/trac/line_bin_2viff -fi 0 -t 0 ");
strcat(to_viff, "-r 320 -c 499 -m 1 -n 1 -b 1 -d 0 -h 0 -s 0");

system(to_ascii);


if((fp1 = fopen("/tmp/trac/line_bin_indata", "r"))==NULL)
{
  printf("Error opening /tmp/trac/line_bin_indata!\n");
  exit(1);
}

if((fp2 = fopen("/tmp/trac/line_bin_outdata", "w"))==NULL)
{
  printf("Error writing to /tmp/trac/line_bin_outdata!\n");
  exit(1);
}

for(k=0; k<13; k++)   fgets(string, 128, fp1);
for(k=0; k<320; k++)
{
  /*--------------------------------------*/
  /* Average the first 8 pixels of a line. */
  /*--------------------------------------*/
  for(r=0; r<8; r++)
  {
    l = 0;
    if((fgets(string, 128, fp1))==NULL)
    {
      printf("Error reading from /tmp/trac/in_test_data!\n");
      exit(1);
    }
    temp = string[l];
    while(temp!='=')
    {
      l = l + 1;
      temp = string[l];
    }
    l = l + 1;

    m = 0;
    temp = string[l];
    while(temp!='\n')
    {
      term2[m] = temp;
      m = m + 1;
      l = l + 1;
      temp = string[l];
    }
    term2[m] = '\0';
    value = atoi(term2);
    num_array[r] = value;
    total = total + value;

    if(upper==1)
    {
      fprintf(fp2, "%d\n", upper);
    }
    else
    {
      fprintf(fp2, "%d\n", value);
```

```
    }
    if((value - max)>0)  max = value;

  }

  /*-------------------------------------*/
  /* Binarize each pixel base on the     */
  /* average of the previous 8 pixels.   */
  /*-------------------------------------*/
  for(r=0; r<491; r++)
  {
    if((fgets(string, 128, fp1))==NULL)
    {
      printf("Error reading from /tmp/trac/in_test_data!\n");
      exit(1);
    }
    l = 0;
    temp = string[l];
    while(temp!='=')
    {
      l = l + 1;
      temp = string[l];
    }
    l = l + 1;

    m = 0;
    temp = string[l];
    while(temp!='\n')
    {
      term2[m] = temp;
      m = m + 1;
      l = l + 1;
      temp = string[l];
    }
    term2[m] = '\0';
    value = atoi(term2);

    average = ((float) total) / 8.0;
    average = (average * factor) + average;
    if(((float) value - average)>0)
    {
      number = upper;
    }
    else
    {
      number = lower;
    }
    fprintf(fp2, "%d\n", number);

    total = 0;
    for(m=0; m<7; m++)
    {
      num_array[m] = num_array[m + 1];
      total = total + num_array[m];
    }
    total = total + value;
  }
}
fclose(fp1);
fclose(fp2);

system(to_viff);

if((upper - max)>0) max = upper;
if(upper==1)
```

```
    {
      max = 1;
    }

    strcpy(to_byte, "vconvert -i /tmp/trac/line_bin_2viff -o ");
    sprintf(string, "%s -n %d", outfile, max);
    strcat(to_byte, string);
    strcat(to_byte, " -t byte -b 0");

    system(to_byte);
    return;
}
```

B.3 COR.C

```
/*****************************************************************/
/* COR.C                                                         */
/*                                                               */
/*            AIR FORCE INSTITUTE OF TECHNOLOGY                  */
/* TRACKS TARGET USING THE TECHNIQUE OF ADAPTIVE CORRELATION     */
/*                  by Capt Dennis A. Montera                    */
/*                       October 15, 1992                        */
/*                                                               */
/* This program executes the correlation plane postprocessing    */
/* algorithm created during my thesis on individual pictures.    */
/* This program does not conduct the actual correlation or       */
/* template autocorrelation.  This program only conducts the     */
/* correlation peak height/shape matching.                       */
/*****************************************************************/


#include <stdio.h>
#include <string.h>
#include <math.h>

main()
{
   /*---------------------------------*/
   /* Define program variables and    */
   /* check for input directory path. */
   /*---------------------------------*/
   FILE *fp1;
   char path[128], term1[25], tmpath[128], string[128], skip;
   int i = 0, j, xx, yy, tmpx, tmpy, count, xcen, ycen, cxx, cyy;
   int fact, txx, tyy;
   float cortemp[7][7], normt, cordata[128][128], holder2;
   float normc, norm1, norm2, norm3;
   double percent, holder;

   /*-------------------------------------*/
   /* Determine which sequence to display */
   /*-------------------------------------*/
   printf("Enter path to be used \n");
   for(i=0;(skip=getchar()) != '\n'; ++i){
     tmpath[i] = skip;
   }
   tmpath[i] = '\0';

   strcpy(path, tmpath);
   sprintf(term1, "_tem.dat");
   strcat(path, term1);


   if((fp1 = fopen(path, "r"))==NULL)
   {
     printf("Error opening %s!\n", path);
     exit(1);
   }

   fgets(string, 128, fp1);

   normt = 0;
   for (yy = 0; yy <= 6; ++yy) {
     for (xx = 0; xx <= 6; ++xx) {
```

```
    if((fscanf(fp1, "%f", &cortemp[xx][yy]))==NULL)
    {
      printf("Error reading from %s!\n", path);
      exit(1);
    }
  normt = normt + (cortemp[xx][yy] * cortemp[xx][yy]);
  }
  fscanf(fp1, "%*s");
}
fclose(fp1);

strcpy(path, tmpath);
sprintf(term1, "_sc.dat");
strcat(path, term1);


if((fp1 = fopen(path, "r"))==NULL)
{
  printf("Error opening %s!\n", path);
  exit(1);
}

fgets(string, 128, fp1);

for (yy = 0; yy <= 127; ++yy) {
  for (xx = 0; xx <= 127; ++xx) {

    if((fscanf(fp1, "%f", &cordata[xx][yy]))==NULL)
    {
      printf("Error reading from %s!\n", path);
      exit(1);
    }
  }
  fscanf(fp1, "%*s");
}
fclose(fp1);


tmpx = 0;
tmpy = 0;
percent = 100.00;
for (xx = 0; xx <= 120; ++xx) {
  for (yy = 0; yy <= 120; ++yy) {
    holder = 0.00;
    holder2 = 0.00;
    count = 0;
    normc = 0.00;
    xcen = xx + 3;
    ycen = yy + 3;

    if(cordata[xcen][ycen] > (cortemp[3][3] * 0.25))
    {

      for (cxx = 0; cxx <= 6; ++cxx) {
        for (cyy = 0; cyy <= 6; ++cyy) {
          txx = cxx + xx;
          tyy = cyy + yy;
          normc = normc + (cordata[txx][tyy] * cordata[txx][tyy]);
          norm1 = cordata[txx][tyy];
          norm2 = cortemp[cxx][cyy];
          norm3 = cortemp[3][3];
          fact = 1;
          if(cxx==3) fact = fact * 8;
          if((cxx==2) || (cxx==4)) fact = fact * 4;
          if((cxx==1) || (cxx==5)) fact = fact * 2;
```

```
            if(cyy==3) fact = fact * 8;
            if((cyy==2) || (cyy==4)) fact = fact * 4;
            if((cyy==1) || (cyy==5)) fact = fact * 2;
            if((cxx==3) && (cyy==3)) fact = fact * 7;

            holder2 = holder2 + (((float)fact) * ((norm1 - norm2) * (norm1 - norm2)));
          }
        }
        holder = ((sqrt((double)holder2)) / ((double)norm3));

        if((count==0) && (holder < percent))
        {
          tmpx = xx + 3;
          tmpy = yy + 3;
          percent = holder;
        }


      }

    }
  }
printf("%s is located at %d,%d \n", tmpath, tmpx, tmpy);


}
```

Bibliography

1. Bar-Shalom, Y. and Tse, E. "Tracking in a cluttered environment with probabilistic data association," *Automatica* 11:451-460 (1975).
2. Birmimal, K. and Bar-Shalom, Y. "On tracking a maneuvering target in clutter," *IEEE Transactions on Aerospace and Electronic Systems* AES-20:635-645 (Sep 1990).
3. Chen, H.-F. and Guo, L. "Continuous-time stochastic adaptive tracking - robustness and asymptotic properties," *SIAM Journal on Control and Optimization* 28:513-527 (May 1990).
4. Fleisher, Michael., et al. "Target Location Measurement by Optical Correlators: a Performance Criterion," *Applied Optics,* 31:230-234 (Jan 1992).
5. Gara, Aaron D. "Real-time Tracking of Moving Objects by Optical Correlation," *Applied Optics,* 18:172-174 (Jan 1979).
6. Gregory, Don A., et al. "Optical Correlator Tracking Nonlinearity," *Applied Optics,* 26:192-194 (Jan 1987).
7. Hutchins, R. G. and Sworder, D. D. "Image fusion algorithms for tracking maneuvering targets," *Journal of Guidance Control and Dynamics* 15:175-184 (Jan/Feb 1992).
8. Kumar, B.V.K. Vijaya "Tutorial Survey of Composite Filter Designs for Optical Correlators," *Applied Optics,* 31:4773-4801 (10 Aug 1992).
9. Khoros Release: 1.0. The Khoros Group, Department of Electrical and Computer Engineering, University of New Mexico, Albuquerque, New Mexico.
10. Law, Paul D. *Correlation Based Distortion Invariant Infrared Tracking.* MS thesis, AFIT/GE/ENG/91D-35. School of Engineering, Air Force Institute of Technology (AU), Wright-Patterson AFB OH, December 1991 (DTIC number not available at this time).
11. Nagarajan, V., et al. "An algorithm for tracking a maneuvering target in clutter," *IEEE Transactions on Aerospace and Electronic Systems* AES-20:560-573 (Sep 1990).
12. Schils, G. F. and Sweeney, D. W. "Optical processor for recognition of three-dimensional targets viewed from any direction," *Journal of the Optical Society of America* A5:1309-1321 (1988).
13. Tam, Eddy C., et al. "Autonomous Real-time Object Tracking with an Adaptive Joint Transform Correlator," *Optical Engineering,* 29:314-320 (Apr 1990).

14. Troxel, Steven E. *Position, Scale, and Rotation Invariant Target Recognition Using Range Imagery.* MS thesis, AFIT/GEO/ENG/87D-3. School of Engineering, Air Force Institute of Technology (AU), Wright-Patterson AFB OH, December 1987 (AD-A188828).

15. Walsh, T. and Giles, M. "Statistical filtering of time-sequenced peak correlation responses for distortion-invariant recognition of multiple input objects," *Optical Engineering* 29:1052-1064 (1990).

16. Uhlmann, J. K. "Algorithms for multiple target tracking," *American Scientist* 80:128-141 (Mar/Apr 1992).

17. Von, Greg, Senior Engineer. Telephone interview. Southern Research Technologies, Inc., Birmingham AL, 16 April 1992.

18. Yee, M. and Casasent, D. "Multitarget data association using an optical neural network," *Applied Optics* 31:613-624 (Feb 10 1992).

Vita

Captain Dennis A. Montera was born on 5 May 1966 in Downey, California. He grew up throughout the Los Angeles, California area, and graduated from John F. Kennedy High School in 1984. He subsequently attended the United States Air Force Academy where he graduated on 1 June, 1988 with a Bachelor of Science in Electrical Engineering. He was commissioned into the United States Air Force on that same day. He married the former Miss Ilona A. Meinberg before reporting to his first assignment at Los Angeles AFB, California. There he served as a Project Engineer for the Deputy Chief of Staff, Developmental Planning until entering the Air Force Institute of Technology in May 1991.

Permanent address: 5141 La Luna Drive
La Palma, California 90623

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	December 1992	Master's Thesis

4. TITLE AND SUBTITLE
Object Tracking Through Adaptive Correlation

5. FUNDING NUMBERS

6. AUTHOR(S)
Dennis A. Montera

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Air Force Institute of Technology, WPAFB OH 45433-6583

8. PERFORMING ORGANIZATION REPORT NUMBER
AFIT/GEO/ENG/92D-07

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Mr. Jim Leonard
WL/AARA, WPAFB, OH 45433

10. SPONSORING / MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION / AVAILABILITY STATEMENT
Approved for public release; distribution unlimited

12b. DISTRIBUTION CODE

13. ABSTRACT (Maximum 200 words)

This paper discusses the use of a correlation based system to track an object through a series of images based on templates derived from previous image frames. The ability to track is extended to sequences which include multiple objects of interest within the field of view. This is accomplished by comparing the height and shape of the template autocorrelation to the peaks in the correlation of the template with the next scene. The result is to identify the region in the next scene which best matches the designated target. In addition to correlation plane postprocessing, an adaptive window is used to determine the template size in order to reduce the effects of correlator walk-off. The image sequences used were taken from a Forward Looking Infrared (FLIR) sensor mounted onboard a DC-3 aircraft. The images contain a T-55 tank and both an M-113 and a TAB-71 armored personnel carrier moving in a columnized formation along a dirt road. The goals of this research were to 1) track targets in the presence of other, and sometimes brighter, targets of similar shape; 2) to maintain small tracking errors; and 3) to reduce the effects of correlator walk-off.

14. SUBJECT TERMS	15. NUMBER OF PAGES
correlation, adaptive template, tracking	162
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	UL